Sem vložte zadání Vaší práce.

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra softwarového inženýrství

Bakalářská práce

ETL server pro potřeby datového skladu fakulty

Radim Lenger

Vedoucí práce: Ing. Stanislav Kuznetsov

5. května 2015

Poděkování

Děkuji Ing. Stanislavu Kuznetsovi za odborné vedení, za pomoc a rady při zpracování této práce. Dále bych tímto chtěl poděkovat všem příbuzným a blízkým za pochopení a podporu.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 5. května 2015 .....

Odkaz na tuto práci

LENGER, Radim. *ETL server pro potřeby datového skladu fakulty.* Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2015.

Abstrakt

Práce se zabývá návrhem a implementací ETL serveru pro potřeby školního datového skladu. V současnosti jsou ETL transformace a úlohy spouštěny ručně správcem a není zaveden konkrétní proces kontroly celého průběhu spouštění ETL transformací. Navržený server tedy zajišťuje pravidelný běh ETL transformací, sleduje jejich činnost a zaznamenává informace o jejich běhu. Zároveň jsem navrhl procesní workflow celé procedury. Server také posloužil pro otestování nasazení historizace a metadat v rámci ETL procesů. Pro pilotní nasazení jsem vybral linuxový server a open source ETL nástroj Pentaho Kettle.
V první teoretické části jsem seznámil čtenáře se základními pojmy a provedl rešerši dostupných řešení ETL procesů. Poté jsem v rámci klíčových ukazatelů výkonnosti sepsal jak měřit a testovat ETL úlohy a jejich běh na serveru. Ve druhé praktické části jsem provedl analýzu současného řešení, navrhl a následně implementoval pracovní workflow ETL serveru. Využil jsem k tomu daemona navrženého v C++ a shell skriptů. Pilotní nasazení jsem poté vyzkoušel a otestoval dle teoretické části a přidal další testy samotného daemona a skriptů.
Při ostrém nasazení bude jistě třeba ještě doladit definici skriptů, jedná se však spíše o drobné úpravy, které nelze vytvořit jinak než s použitím ostrých dat.

Klíčová slova datový sklad, ETL, Pentaho, Kettle, C++, ETL server, historizace, metadata, BI

Abstract

The bachelor thesis deals with the design and implementation of ETL server for the school data warehouse purposes.

Nowadays the ETL transformations are being started manually by admin where there is no specific process control. Designed server provides regular running of ETL transformation, monitors their activity and records information of running ETL scripts. I also designed process workflow of the entire procedure. Server also serves as testing machine for pilot deployment of historization and metadata within the ETL processes. I've chosen linux server and the open source tool Pentaho Kettle.
In the first theoretical part the reader is informed about basic therms. I've also made a search report of available solutions for ETL processes. Concurrently I wrote down the measurement and testing of ETL jobs and their running on the server within the key proccess indicators section. In the second practical part I've done analysis of current solution. Thereafter I've designed and implemented ETL server workflow. I've prepared daemon written in C++ and I also wrote a few shell scripts. In conclusion I've tested the server regarding to the theoretical part and I've added some tests of my daemon and scripts.
Within the real deployment admin will need to change and harmonize shell scripts regarding to the real data sets.

Keywords data warehouse, ETL, Pentaho, Kettle, C++, ETL server, historization, metadata, BI

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Společnosti a instituce v dnešní době vlastní velké objemy dat, se kterými je třeba rozumně nakládat a zpracovávat je. Proto vznikla myšlenka datových skladů - spojení snadného přístupu k datům a následné reportování informací ve prospěch společností. Datové sklady byly spíše doménou větších společností, dnes se však díky rozvinutým technologiím dostávají i do menších firem a v našem případě i do škol. Na Fakultě informačních technologií máme také vlastní datový sklad, který byl vytvořen v rámci diplomové práce. Tento sklad se však musí potýkat s poměrně složitým přistupem k datům. Data nejsou získávána automaticky a vždy je správce závislý na nepravidelných exportech. Sklad zároveň poskytuje mnoho možností jak vylepšit současné řešení jednotlivých procesů, které jsou v rámci datového skladu vykonávány.
Principem datového skladu je možnost slučovat a využívat data z mnoha rozličných zdrojových systémů. Jednotlivé zdroje s sebou samozřejmě nesou rozličný formát a způsob práce s daty. Z toho vyplývá nutnost umět data nejen načíst, ale také vyčistit a připravit pro použití v rámci struktury datového skladu.
Současné řešení fakultního datového skladu funguje na bázi pravidelných exportů za určité časové období do textových souborů a následného ručního spouštění předpřipravených ETL transformací. S rostoucími požadavky na frekvenci nahrávání dat ale současné řešení přestává být dostatečně flexibilní a neposkytuje komplexní informace o průběhu jednotlivých ETL procesů. Vznikla tedy potřeba celý proces zautomatizovat. Součástí této automatizace je vytvoření samostatného serveru, který bude ETL transformace zajišťovat a provádět jejich dokumentaci a kontrolu.
ETL server tedy oproti současnému řešení umožní aktualizovat data v relevatně zvoleném časovém úseku a poskytne aktuální data pro tvorbu reportů. Tím zajistí informovanost pro správce skladu v rámci provedených změn a původu jednotlivých dat a zároveň umožní snazší aktualizaci dat pro datový sklad.
V první části své práce seznámím čtenáře s terminologií a principem ETL pro-

cesů a zavedu důležité pojmy. Poté provedu rešerši současných ETL nástrojů a porovnám jednotlivá řešení v rámci možností a vhodnosti použití pro fakultní datový sklad. Zaměřím se hlavně na dostupnost a efektivitu jednotlivých nástrojů a následně vyberu nejvhodnější řešení. Ve druhé části se zaměřím na samotnou implementaci ETL serveru. Na serveru připravím automatické a sledované spouštění jednotlivých procesů. Server bude dále poskytovat prostor pro historizaci, kterou se zabývá ve své bakalářské práci Robert Kotlář[4] a zaznamenávání metadat - metadata připravuje ve své bakalářské práci Jakub Krejčí[5]. Na závěr provedu otestování nasazeného řešení a zhodnotím možnosti do budoucna.

Cíl práce

Cílém mé práce je provést popsání problematiky ETL procesů, vytvoření rešerše trasnformačních nástrojů a navrhnutí ETL serveru. V rámci návrhu bude třeba vytvořit automatizovaný a kontrolovaný proces a navrhnout způsob kontroly běhu serverových aktivit. ETL server bude následně podroben testování jeho funkcionality.

Část I

Teoretická část

Kapitola 1

Zavedení pojmů

1.1 Datové sklady

Datové sklady v současné době zažívají velký nástup, a proto vzniká mnoho nových technik, jak sklady navrhovat a stavět. Přesto však tyto návrhy vycházejí ze dvou hlavních přístupů. S těmito definicemi přišli Bill H. Inmon a Ralph Kimball[6]. V následující kapitole se pokusím oba způsoby přiblížit a seznámit čtenáře s funkcionalitami.

1.1.1 Datový sklad dle Inmonova návrhu

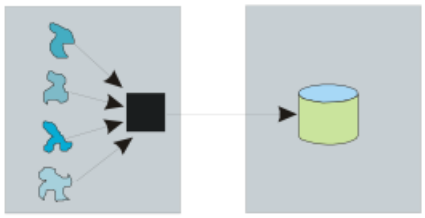
Datový sklad byl poprvé definován Billem H. Inmonem, který je považován za "otce"datových skladů, jako subjektové zaměřený, integrovaný, stálý a časově rozlišitelný soubor dat, který podporuje volby a potřeby managementu.[2].

- **Subjektová orientace** - data jsou tříděna dle jejich smyslu a typu - nikoliv dle jednotlivých zdrojů a systémů, ze kterých pochází.
- **Integrovanost** - jedná se o propojení v rámci celého podniku - nezáleží z jaké části podniku či z jakého oddělení data pochází - jsou skladována na jednom místě.
- **Stálost** - datový sklad slouží "pouze ke čtení"- data jsou do něj přidávána automaticky, není možné z něj mazat, data v něm měnit či svévolně přidávat - přidávání je dáno specifickým procesem.
- **Časová rozlišitelnost** - rozlištelnost je zajištěna souhrnem dat nejen současných, ale i dat z minulosti. Následné vnitřní struktury nám umožňují na data nahlížet v časovém průběhu a to za velmi nízké náklady (samozřejmě však záleží na vnitřní implementaci).

Dle zásad pana Inmona se sklad dělí na čtyři sekce. Data jsou získávána v rámci sekce Source systems. Poté se naskýtá několik možností, jak zpracovávat získaná data - setkáváme se s těmito přístupy[1].

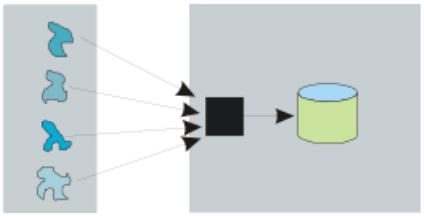
- **Data zpracována na stroji, odkud pochází** - tento způsob má tu výhodu, že data jsou vždy okamžitě přístupná pro ETL procesy. Převažují však nevýhody. ETL procesy jsou velmi náročné na výkon hostujícího stroje. Na tomto stroji nemusí být vždy dostatek operačního času pro naše ETL procesy - nejsme tedy schopni plného řízení a nemáme kompletní kontrolu na spouštěním. Vzhledem k tomu, že stroj má primárně jiný účel, naše ETL procesy budou vždy až na druhém místě.

Obrázek 1.1: Data zpracována na stroji, odkud pochází. Zdroj [1]



- **Data zpracována na stroji s datovým skladem** - v případě umístění na stroji s datovým skladem máme výhodu v úplné kontrole nad během ETL procesů. Velkou výhodou také je, že po zpracování dochází k přesunu transformovaných dat do skladu velmi rychle a bezpečně. Dalším plusem shledáváme možnost získání více strojových/hardware zdrojů v případě nutnosti. Mezi nevýhody se řadí nutnost přenášet mnoho raw dat[1] - nahrání obrovského množství raw dat nás může stát mnoho času. Další nevýhodou je nemožnost přístupu k dodatečným zdrojům dat - po přesunu raw dat jsme omezeni na to, co jsme dostali. Oproti prvnímu způsobu již nemáme přístup k doplňujícím informacím.

Obrázek 1.2: Data zpracována na stroji s datovým skladem. Zdroj [1]

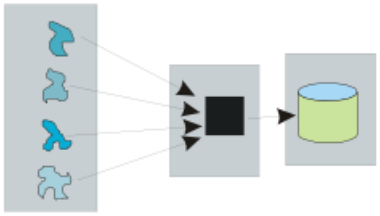


- **Data zpracována na samostatném stroji** - hlavní výhodou totoho přístupu je, že stroj určený pro ETL procesy je výhradně používán pouze pro transformace. Tím pádem nedochází k nutnosti střídání s dalšími aplikacemi či procesy a máme možnost využití zdrojů na maximum. Jedná

[1] Jedná se o dosud nezměněná data ze zdrojových systémů.

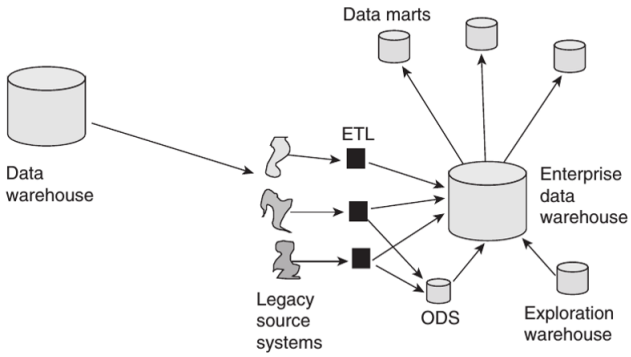
se tedy o nejefektivnější přístup jak provádět ETL trasnformace. V případě, že máme zajištěn rychlý přístup ke zdrojovým datům a možnost snadno odeslat transformovaná data do cílové systému, provádíme ETL procesy velmi rychle. Mezi nevýhody se řadí, jako v případě umístění na cílovém stroji, nutnost přesunu objemných raw dat a přibývá také potřeba odesílat trasnformovaná data na stroj s datovým skladem.

Obrázek 1.3: Data zpracována na samostatném stroji. Zdroj [1]



Poté jsou data uskladněna v jediném centralizovaném datovém skladu. Další možností je sekce pro předzpracování dat v Operation data store (zde jsou data trasnformována ETL procesy a připravena pro rychlejší použití) v sekci Staging area. Poslední sekcí jsou data marty - jednotlivé databáze upravené dle potřeby oddělení firmy či instituce. Pro názornost je přiložen obrázek 2.1. S touto architekturou seznamuji i z důvodu, že současný fakultní datový sklad bude zřejmě postupně přetvořen dle tohoto návrhu. Konkrétní důvody zmíním dále ve své práci, až bude čtenář seznámem s návrhem dle Kimballa.

Obrázek 1.4: Architektura datového skladu dle Inmonovy definice. Zdroj [2]

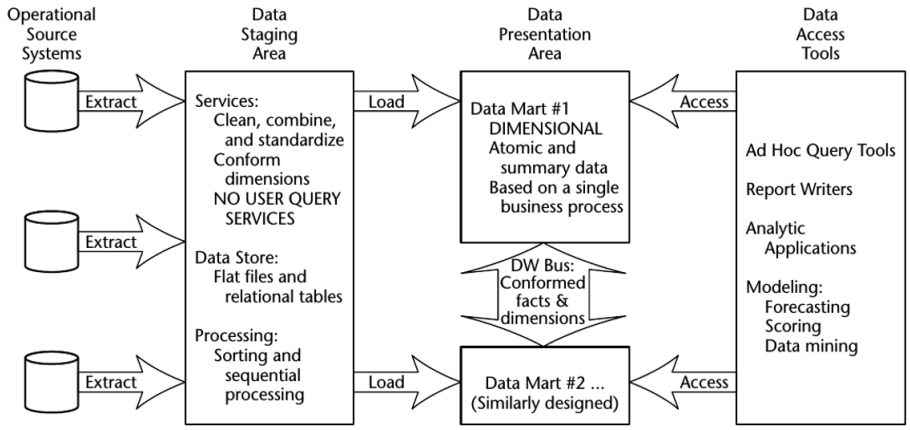


1.1.2 Datový sklad dle Kimballova návrhu

Další velmi významnou osobou v prostředí datových skladů je již zmiňovaný Ralph Kimball. Dle jeho návrhu je datový sklad postaven na čtyřech hlavních komponentách.[3]

- **Source systems** - do source systems patří všechny systémy, které slouží jako datové zdroje pro datový sklad. Tato část návrhu by neměla být vnímána jako součást datového skladu, neboť pravděpodobně máme velmi malou nebo žádnou kontrolu nad obsahem a formátem dat v těchto systémech.
- **ETL system nebo také někdy Staging area** - v této pracovní oblasti se provádí všechny ETL transformace a případné podpůrné procesy. Tato oblast zahrnuje vše, co se nachází mezi zdrojovými systémy a prezentační oblastí.
- **Data presentation area** - v této oblasti jsou data organizována, uložena a zpřístupněna pro přímé dotazování od uživatelů či uživatelských aplikací. V této oblasti se nachází databáze datového skladu - data marts.
- **BI applications či Acces Tools** - tato oblast by měla být také vnímána jako vnější součást datového skladu. V této oblasti se nachází všechny nástroje pro zobrazení dat a jejich využívání, které mají do datového skladu přístup.

Obrázek 1.5: Architektura datového skladu dle Kimballovy definice. Zdroj [3]



Návrhy se, co se týče sekcí, mohou zdát velmi podobné, ovšem v rámci stavby datového skladu a rozložení jednotlivých komponent je na první pohled vidět mnoho rozdílů a jejich dopadů. V rámci Kimballa není nutné tvořit jeden komplexní a složitý datový sklad, ale můžeme vytvořit několik implementačně snazších data martů. Také zde vidíme rozdíl v sekci, která se stará o ETL procesy. U Inmona můžeme vidět "databázový"přístup pro uchování transformací a změn, kdežto u Kimballa proudí transformovaná data přímo do jednotlivých marketů, které mohou data rovnou distribuovat mezi uživatele. Nejzásadnějším rozdílem je však iterativní přístup v případě Kimballa. Jestliže

se nesetkáváme s plnou podporou, můžeme v tomto případě za relativně krátký časový úsek nabídnout uživatelům výstupy. V případě návrhu dle Inmona musí uživatel čekat velmi dlouho a pokud není stoprocentně přesvědčen o datovém skladu, je tento přístup velmi těžké prosadit. Zmínil jsem dle mého názoru ty nejdůležitější rozdíly v rámci rozsahu mé práce, ale obecně je v obou přístupech mnoho rozdílů a každý má své klady i zápory.

1.1.3 Shrnutí

Datový sklad fakulty je vystavěn na principech Ralpha Kimballa. Hlavním důvodem byl právě iterativní přístup. V rámci fakulty se jednalo o prvotní nasazení, a proto nebylo možné datový sklad několik let budovat bez zajištění určitých výsledků. Přesto jsem se rozhodl zařadit definici datového skladu od Inmona, neboť mi přijde velmi srozumitelná a přesná. Zároveň se po pilotním nasazení skladu na principech Kimballa uvažuje přestup na Inmonův návrh. Inmonův návrh je robustnější, snazší na údržbu a rozšiřování. Ovšem je také, jak již bylo zmíněno, mnohem náročnější na vystavění. V současné době máme však již velmi dobré výstupy ze skladu a dobrou zpětnou vazbu - díky tomu je možné uvažovat o přestavbě. Návrh a design datových skladů je samozřejmě velmi komplexní záležitostí. V předchozích sekcích jsem se snažil čtenáři přiblížit rozličné přístupy a také zadefinovat typ datového skladu, pro který budu tvořit ETL server. Celkové problematice návrhu a rozdílnosti v přístupech se blíže věnuje diplomová práce Stanislava Kuznetsova[7], kde lze navíc nalézt detailní popis implementace datového skladu na fakultě informačních technologií. Pro mou práci je podstatné, že ETL server je nutnou součástí návrhu dle Inmona, ale zároveň má velké využití v případě Kimballa. V našem případě se bude dát použít pro oba způsoby.

1.2 ETL - definování pojmu

Pojem ETL je zkratka ze slov **Extract, Transform a Load**. Volně přeloženo se tedy jedná o získání, modifikaci a nahrání/uložení. ETL procesy patří v rámci návrhu a implementace k jedné z nejpodstatnějších částí tvorby datového skladu. Používají se však i v rámci běžných databází, či pro exporty dat. Hlavním přínosem transformací pro datové sklady je zajištění integrovanosti. Díky ETL procesům jsme schopni nahrát data z různých zdrojů a udržovat jejich čistotu. Data se samozřejmě musí modifikovat a také musí být pročištěna, abychom byli schopni uživatelům datového skladu podávat plně relevantní data. V neposlední řadě zajišťují stálost - pravidelným spouštěním ETL procesů hlídáme aktuálnost dat a zároveň poskytujeme možnost nahlížet na data přes časovou osu.

1.2.1 Extract

Pod slovem Extract tedy v rámci datových skladů vnímáme získání dat z různých zdrojových systému. Tyto systémy mohou mít v podstatě libovolný výstup. Může jednat i o přímé napojení - sálové počítače, relační databáze, XML zdroje, flatfile (csv, xls), ERP systémy a webové logy [8]. Data se také kromě rozdílného formátu mohou vyskytovat i na různých platformách a díky ETL procesům/nástrojům jsme tak schopni nezávisle na platformě tato data spojovat a seskupovat na jedno místo. Odpadá nám tedy starost jak slučovat data z platformně závislých zdrojů (např. MSSQL[2] a PostgreSQL[3]). Při návrhu je tedy v tomto kroku nutné analyzovat typy zdrojů, jejich dostupnost a zajistit dobrý přístup. V případě velkého objemu dat by mohlo při exportu dojít k vyššímu zatížení serveru, což by nemuselo být žádoucí v rámci zpracovávání běžných činností. Dále je nutné pro ETL procesy/nástroje zajistit vhodnou a bezpečnou dostupnost.

1.2.2 Transform

Transform pod sebou skrývá v podstatě jakoukoliv úpravu, kterou na získaných datech provedeme. Obyčejně se však rozděluje na dvě základní části - vyčištění a úpravu dat. Trasnformace našich dat bude vždy velmi záviset na datových zdrojích - v případě kvalitních zdrojů budou naše trasnformace snazší, jelikož nebude třeba složitého čištění a nutnosti provádět úpravy v datech. Při plánování je tedy nutné zvážit, na jaké úrovni chceme data měnit - zda je pro nás možné vyčistit data na jejich zdrojové úrovni a nebo zda se o to postarají ETL transformace. Zároveň je třeba přesně definovat, jak nakládat s daty při jejich úpravě - jakým způsobem můžeme řešit nekonzistence, jaké jsou mezi jednotlivými položkami vazby a jaké změny můžeme dělat například v rámci datových typů.

1.2.2.1 Vyčištění

Vyčištěním rozumíme odstranění nevalidních záznamů. Může se tak jednat o nevyplněné sloupce, prázdné hodnoty a nebo přímo špatně vyplněná data. V rámci databází se často setkáváme právě s prázdnými sloupci, neboť podmíněnost správného datového typu je většinou nastavena. V běžných souborech se však můžeme setkat i s přímo špatně vyplněnou hodnotou - díky transformaci jsme schopni nevalidní položky odfiltrovat a nebo si je upravit.

1.2.2.2 Úprava

Úprava dat v rámci ETL procesů znamená nejen validace získaných dat, ale i transformace v rámci návrhu cílového systému. Naším cílem je zdrojová data

[2]Microsoft SQL Server založený na TransactSQL.
[3]Objektově relační databázový systém založený na SQL.

opravit či upravit na cílový tvar a vlastnosti. Můžeme tak snadno například měnit typy jednotlivých položek, pomocí regulárních výrazů opravit chyby vzniklé zadáváním uživateli a nebo si data přetransformovat do námi kýženého výstupu (spojení více buněk do jedné, apod.)

1.2.3 Load

Finálním krokem v rámci ETL procesů je Load, tedy nahrání/uložení. V tomto kroku máme již připravena vyčištěná a upravená data a můžeme je zavést do cílového systému. Většina ETL nástrojů má tento proces zvládnutý velmi kvalitně a nedochází k nadměrnému zatěžování datového skladu/cílového systému. Zde však velmi záleží, zda používáme ETL nástroj, nebo zda si řídíme samostatně ETL procesy vlastními programy/skripty.

ETL procesy jsou komplexním souborem procesů, a proto existujte několik způsobů jak je vytvářet a jak měřit jejich efektivitu - těmto tématům se budu blíže věnovat v další části své práce. Také přístupy jak s ETL procesy nakládat se mohou lišit. V rámci datových skladů se běžně používá nahrávání dat po jednotlivých dávkách - v praxi však můžeme potřebovat aktuální data a v tu chvíli bychom o ETL procesech měli uvažovat jako o průběžném toku dat ze zdrojů do cílového systému. Naším hlavním úkolem je co nejmenší ovlivnění a zatížení jak zdrojových systémů, tak i cílových. Měli bychom tedy proces ETL co nejvíce optimalizovat [8].

Kapitola 2

Tvorba ETL procesů

2.1 ETL nástroj versus vlastní programování

Při stavění datového skladu bychom si měli položit otázku, zda využít ETL nástroj a nebo zda si budeme chtít ETL transformace programovat. Nelze na ni však odpovědět jednoznačně, neboť vždy závisí na konkrétním případu a nemůžeme pevně určit, že ETL nástroj či vlastní naprogramování je jediným možným řešením. Předtím, než dojdeme k rozhodnutí musíme důkladně projít požadavky na ETL procesy a zároveň si určit potřeby a cíle. V následujících bodech se pokusím shrnout myšlenky z článku Garyho Nissena [9], který za pomoci R. Kimballa velmi trefně shrnul body, které by nám měly pomoci při rozhodování a doplním dle mého názoru další relevatní body.

2.1.1 Klady ETL nástroje

- **Jednoduchost a snadnost** - technicky schopní uživatelé, kteří dobře rozumí business logice datového skladu, jsou schopni se naučit v ETL nástroji a sami se podílet na tvorbě ETL procesů. Tím nám odpadá nutnost řešit porozumění programátora a designéra a zároveň nám to poskytuje větší variabilitu.
- **Metadata** - většina současných nástrojů rovnou v základu poskytuje možnost sledování metadat - ty jsou v rámci BI [4] velmi důležité a umožňují nám vypátrat, odkud data přišla a jakým procesem prošla. Zároveň ETL nástroje nastolily většinu pravidel v rámci metadat (pro datové sklady), která jsou po vývojářích stejně vyžadována.
- **Exporty informací o procesech** - nástroje také nabízejí snadné provádění výpisů již zmíněných metadat, logování důležitých informací a výpisy o průběhu ETL transformací. Odpadá nám tedy nutnost tvořit složitě reporty nad jednotlivými funkcionalitami.

[4] Business Intelligence - dovednosti, znalosti a postupy používané v podnikání.

- **Možnost napojení rozličných zdrojů a cílů** - současné nástroje jsou velmi agilní hlavně v přístupu ke zdrojům. V případě, že tvoříme datový sklad závislý na mnoha rozličných datových zdrojích, nám ETL nástroje poskytují velmi rozsáhlé možnosti. Stejně tak můžeme snadno volit cíle našich ETL transformací, a tak není problém data stejně tak distribuovat do mnoha dalších cílových systémů.

- **Výkon** - ETL nástroje poskytují (samozřejmě zde závisí na konkrétních nástrojích) velmi kvalitní výsledky v rámci výkonu. V případě, že se zaměřujeme na velké objemy dat, nám může ETL nástroj ušetřit mnoho systémových požadavků. V tomto případě samozřejmě platí, že open source řešení poněkud zaostávají za placenými verzemi.

Předchozí body čerpaly informace z článku Garyho Nissena. Následující jsem doplnil dle zkušeností s ETL nástroji.

- **Modifikovatelnost** - v tomto případě se jedná o výhodu hlavně open-source řešení (pokud chceme ušetřit náklady). Tato řešení nám poskytují možnost doprogramovat si chybějící funkcionality, a tak můžeme využít základních funkcionalit ETL nástroje a ušetřit na cenném čase programátorů. Ti mohou jen dotvořit chybějící moduly a z našeho nástroje se stane ještě mocnější produkt.

- **Udržitelnost** - u ETL nástrojů je mnohem snazší udržet si dokumentaci a provádět údržby systému. Nejsme tak závislí na dokumentaci od autora naprogramovaných ETL procesů (která může být po letech neaktuální či špatně vedená) a zároveň je snazší zaučit nové pracovníky.

2.1.2 Klady vlastního naprogramování ETL procesů

- **Automatické testy** - vlastním naprogramování můžeme zajistit velmi kvalitní testování našich transoformací. Snadno tak můžeme sledovat chování ETL transformací na různých datových souborech a být velmi rychle informováni o výsledku a případných problémech. V tomto směru nám napomáhá využití kombinace již naprogramovaných testů (jako např. JUnit apod.) v kombinaci se skriptovacími jazyky, které dohromady tvoří velmi silný testovací prostředek.

- **Konzistence** - objektové orientované techniky nám umožňují snadno sjednotit konzistenci v rámci chybových hlášení, ověřování výsledku a aktualizaci metadat. V případě ETL nástroje nemáme často možnost mít jednotné chybové hlášky a to nám ztěžuje hledání chyb v našich transformacích. Stejně tak při ověřování výsledku si můžeme sami naprogramovat jednotné výsledné hlášení pro snazší orientaci.

- **Větší variabilita v metadatech** - nejsme-li vázáni určitými pravidly, jsme schopni si metadata upravit přesně podle našich potřeb a mít tak důkladné a komplexní informace o průběhu jednotlivých operací a změnách v datech.

- **Flexibilita** - vlastnost zmíněna v podstatě i v předchozích bodech. Vlastní naprogramování poskytuje téměř neomezené možnosti. Týká se také náročnosti na obsluhu/tvorbu ETL procesů - ve firmě můžeme případně využít vlastní programátory a tím ušetřit na nákladech za nové pracovníky/zaučení stávajících.

2.1.3 Výběr vhodného řešení na základě uvedených kladů

Po zhodnocení informací uvedených výše jsem si pro svou práci zvolil nasazení ETL nástroje. Vzhledem k fakultním podmínkám si prozatím nemůžeme dovolit tým programátorů, který by programoval ETL procesy a staral se o jejich dokumentaci a následnou podporu. Mé rozhodnutí je ovlivněno i velikostí fakultního datového skladu - jedná se spíše o sklad menších rozměrů v porovnání se sklady běžícími v běžném obchodním prostředí - o to snazší a intiutivnější bude využití běžného ETL nástroje. Z výše uvedených kladů ETL nástrojů oceníme také velmi snadný přístup ke zpracování dat z rozličných zdrojů. V současnosti využíváme hlavně systém KOS, okrajově však také již pracujeme s daty ze systémů Progtest a Úvazkostroj. Každý tento systém má vlastní přístup ke správě dat a odpadne nám tedy náročnost naprogramování složitých procesů. Další výhodou je obsah metadat a historizace, jak jsem zmínil v začátku své práce. Souběžně s mojí prací jsou ve vývoji i práce řešící problematiku metadat a historizaci - zde jim ETL nástroj velmi pomůže a poskytne jim i možnost doprogramovat a dotvořit vlastní moduly a využít již integrovaných procesů. Závěrem bych ještě rád zmínil udržitelnost systému - pro další vývoj bude snadné navázát na pilotní nasazení ETL serveru. Grafické prostředí ETL nástroje umožní snadné nastudování ETL transformací, ze zalogovaných dat zase snadno vyčteme výsledky ETL procesů a jejich chybovost či výkonnost.

2.2 Dostupné ETL nástroje

ETL nástroje můžeme dělit do mnoha kategorií - některé jsou integrovány v rámci velkých systémů, které obsahují další logiku pro datové sklady, jiné jsou zase součástí různých BI řešení, a pak tu máme také standalone aplikace [5], které slouží pouze pro potřeby ETL procesů. Z hlediska mé práce a určení ETL nástroje pro naši fakultu jsem si zvolil rozdělení na placené a neplacené

[5] Aplikace schopné běžet samostatně bez potřeby dalších aplikací/procesů.

nástroje. V následujícím odstavci se pokusím shrnout klady placených a neplacených ETL nástrojů. Poté zhodnotím dvě open-source řešení, z nichž jedno vyberu pro naše potřeby.

2.2.1 Placené ETL nástroje

V této sekci provedu komplexnější rozbor dvou řešení podrobněji.

- **Informatica - Powercenter** - tento nástroj společnosti Informatica patří mezi špičku současných placených nástrojů a v žebříčcích ETL nástrojů je vždy mezi nejlepšími (kupříkladu uvedu nezávislý žebříček ETL Tools Comparison [10]). Tento nástroj lze pro vlastní potřeby stáhnout zdarma - je ovšem omezen počet řádků, které mohou být zpracovány, a verze je to jednouživatelská. Její funkcionalita je dále omezena na zpracování pouze 1 úlohy [6]. Placená verze už není limitována řádky ani uživateli a koupě obsahuje také podporu. Cena je v současné době 8000$ za jednoho uživatele na rok používání [11]. Mezi jeho zápory bylo nejčastěji zmiňováno napojení zdrojů - zde Informatica pokulhává za svými konkurenty.

- **Oracle - Data Integrator** - u tohoto nástroje musím předem deklarovat, že se jedná o mix řešení - je deklarován jako ETL i ELT nástroj. Dle dokumentace je však hlavní informací, že oproti svým konkurentům i v rámci ETL procesů staví na ELT architektuře [12]. To je poměrně zajímavý přístup a měl by přinášet zrychlení procesů - kdy transformační procedury běží již v rámci databáze na datovém skladu. Licence na procesní stroj stojí 30000$ a další doplňky jako Support a Update stojí zhruba 15 840$ [13].

2.2.2 Neplacené ETL nástroje

V sekci neplacených provedu komplexnější rozbor tří řešení, které jsou zmiňovány jako kvalitní ETL nástroje. Poté vyberu nástroj, který použiji pro nasazení na ETL server. Seznam ETL nástrojů jsem udělal jako průnik z několika stránek, které se zabývaly žebříčky top zdarma nástrojů. Viz zdroje [14], [15], [16].

- **Talend** - jedná se o velmi oblíbený nástroj a ETL nástroj je součástí většího BI řešení. Mezi zákazníky společnosti Talend patří například i firmy jako Allianz, Sony, Orange či Lenovo. Kromě ETL nástroje (který je součástí Big Data balíčku) nabízejí další business aplikace - např. Data Quality, Data Integration. Cena v rámci Big Data enterprise edice je bohužel pouze na vyžádání dle specifikací. Enterprise edice poskytuje opět

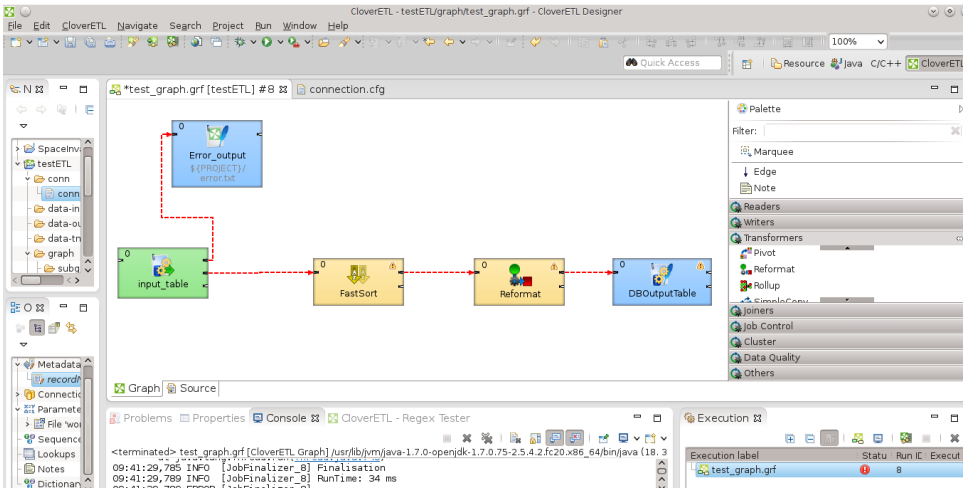
[6] Pojem úloha v této souvislosti značí soubor instrukcí, které se mají vykonat v rámci ETL transformací.

mnohá vylepšení v rámci rychlosti, podpory a integrovaných vylepšení. Tento produkt je dle informací dostupný na všech běžných platformách (Windows, Linux - různé distribuce a MacOS) [17].

- **Pentaho - Kettle** - tento nástroj nabízí společnost Pentaho, která opět dodává kompletní BI řešení. Nástroj Kettle a hlavně jeho součást Spoon, která se stará o ETL procesy, je naprosto intuitivní, snadná na ovládání a její instalace je velmi uživatelsky přívětivá. Tento nástroj se také pro svou funkcionalitu a možnosti používá v rámci předmětu MI-EDW na fakultě informačních technologií. Stejně jako v předchozím případě se jedná o multiplatformní nástroj. Kettle umožňuje snadné připojení na rozličné zdrojové systémy, ať už flatfile či databáze, a to velmi intuitivním nastavením. Poté je již snadné tvořit samotné ETL transformace či plánovat pravidelné úlohy. Nástroj dále umožňuje použití metadat a historizace - to je velmi podstatná součást procesů pro datové sklady. Uživatelské prostředí je velmi příjemné a dobře se v něm pracuje. Další výtečnou funkcionalitou je možnost doprogramovat a doinstalovat si dodatečně vlastní pluginy. Možnosti nejsou samozřejmě neomezené, ale tento přístup velmi rozšiřuje schopnosti uvedeného nástroje.

- **CloverETL** - v rámci instalace se jedná o snadný a intuitivní nástroj a vše probíhá bez problémů - stejně tak napojení na databázi. Horší je již ovšem orientace v rámci tvorby ETL procesů. Celkově se svým uživatelským rozhraním podobá Eclipse a dokonce je možné ho doinstalovat pouze jako doplněk Eclipse, nikoliv jako standalone aplikaci. Pro ETL nástroj mi toto zobrazení nepřišlo příliš šťastné - špatný náhled na dostupné ETL procesní nástroje a složitá orientace. Líbily se mi možnosti, které v rámci transoformací CloverETL nabízí. Z dostupných informací jsou rozdíly mezi community edicí a enterprise edicí, například možnost exportu všech součástí ETL procesů, nikoliv jen grafu. Dále také logovací konzole a použití rychlejšího clover engine. Všechny rozdíly lze naléz ve zdroji [18]. Podařilo se mi dohledat cenu produktu CloverETL Designer, který stojí 5000$, další produkty se opět cení až na dotázání prodejce.

Obrázek 2.1: Jednoduchý ETL proces, který vezme data z tabulky, setřídí je a upraví a poté nahraje zpět.



Závěrem bych rád ještě zmínil problémy při hledání neplacených ETL nástrojů. Obecným problémem je malá informační hodnota webových stránek a dokumentů. Najít rozdíl mezi neplaceným a enterprise řešením je často nemožné (mnohokrát jsem se ho dočetl až na různých diskuzních fórech) a stejně tak i dohledání alespoň předběžné ceny za enterprise produkt. Těžko si pak zákazník bude chtít pořídit produkt, o němž neví, čím je lepší a kolik stojí. Tento neduh mě u takto velkých společností zarazil a pokládám ho za velký nedostatek. Co mě naopak zaujalo, byl velký zájem i ze strany rozsáhlých firem, u kterých bych očekával spíše využití placeného softwaru. Tento přístup mě velmi potěšil a myslím, že může i pro neplacené či open-source řešení přilákat mnoho nových zákazníků a tím i zlepšit jejich kvalitu.

2.3 Výběr ETL nástroje

Z výše uvedených nástrojů jsme se rozhodli zvolit open-source řešení. Toto řešení přinese pro fakultu velkou peněžní úsporu a zároveň nebudeme limitováni licencemi a platností ETL nástroje. V rámci open-source nástrojů jsem dle hodnocení zvolil dva favority na nasazení v rámci fakultního datového skladu - **Pentaho Kettle, CloverETL**. Prvním testovaným produktem byl **CloverETL** - tento nástroj mi přišel funkcionalitami velmi robustní a kvalitní. Nezaujal mě však zpracováním a obecně jsem z něj měl dojem, že se jeho použití hodí spíše mimo prostředí datových skladů. Pro naše použití byla hlavním záporem právě nepřehlednost procesů - v rámci fakulty je vhodné, aby se uživatel byl schopen s prostředím seznámit velmi rychle a snadno a zároveň aby člověk znalý základních pojmů mohl číst transformace a případně navrhovat změny. Druhým testovaným byl **Pentaho Kettle** - tento nástroj

mezi svými konkurenty zvítězil v rámci všech kategorií 2.1. Instalace a napojení na databáze jsou velmi jednoduché a funkční v rámci rozličných platforem (testováno na několika distribucích Linux i na operačním systému Windows). Prostředí je intuitivní, přehledné a procesy jsou zobrazeny dostatečně podrobně. Funkcionalitou dosahuje kvalit všech svých konkurentů a pro naše procesy splňuje všechny požadavky. Jako bonus umožňuje nasazení historizace a zaznamenávání metadat. Další výhodou je znalost prostředí v rámci kolektivu pracujícího na datovém skladu. Jak jsem již zmínil, nástroj je použit i pro předmět MI-EDW, a tak jsou s ním seznámeni nejen cvičící, ale i naši studenti. V tomto směru vidím přínos takový, že bychom mohli do budoucna využít právě studentů, kteří by například v rámci semestru mohli pracovat na zlepšení současného řešení a nebo se i po škole zapojit do práce na školním datovém skladu. Poslední velkou výhodou je kvalitní uživatelská základna - problémy se dají většinou řešit na diskuzních fórech. Kromě uživatelské základny také dobře funguje podpora ze strany společnosti Pentaho - v rámci školního projektu jsem měl možnost pracovat s dalším jejich produktem. Přestože jsme používali verzi zdarma, věnovali se nám zaměstnanci Pentaha a setkali jsme se s velkou ochotou.

Název	Informatica	Oracle DI	CloverETL	Pentaho Kettle
Open source	✗	✗	✓	✓
Přehlednost	✓	✓	✗	✓
Funkcionalita	✓	✓	✓	✓
Modifikovatelnost	✓	✓	✗	✓
Snadná instalace	✗	✗	✓	✓
Multiplatformnost	✓	✓	✓	✓
Zdrojové systémy	✗	✓	✗	✓
Cílové systémy	✓	✓	✓	✓
Podpora	✓	✓	✗	✓
Dokumentace	✓	✗	✓	✓

Tabulka 2.1: Tabulka srovnání ETL nástrojů

2.4 Požadavky na ETL procesy

Důležitou součástí tvorby ETL procesů je jejich validace a kontrola funkčnosti a náročnosti. Pro ETL procesy neexistují předepsané požadavky, které by měly ETL trasnformace splňovat či projít. Obecně se však můžeme opřít v tomto směru o jejich testování. Díky tomu se dá na ETL procesy nahlížet jako na běžný software. V tu chvíli budeme uvažovat výsledky testů jako zhodnocení jejich kvality. Mezi běžná testování tedy patří testování požadavků, validace

dat, integrační testy, poté report testy a následně user acceptance testy [7]. Na závěr bychom neměli opomenout výkonnostní testy a regresní testy, které nám usnadní testování úprav v rámci změn v ETL procesech. Jednotlivé body podrobně rozberu dále v této sekci. Testování a zhodnocení nejen ETL procesů, ale celého datového skladu, by mělo patřit mezi běžné činnosti prováděné při tvorbě a užívání datového skladu. Odlišnost v rámci testování ETL procesů od běžných testů vidím ve velké závislosti na dodaných datech. ETL procesy je třeba testovat na co nejpodobnějších datech, abychom dostali validní informace. V této sekci jsem čerpal informace hlavně z článku společnosti PQA, která se zabývá testováním. [19]

- **Testování požadavků** - na základě této sekce se odvíjí všechny ostatní, je třeba ji neopomenout a provést důkladně. Je důležité zjistit, jaké jsou klíčové požadavky v dané firmě na ETL procesy - např. zda musí být co nejrychlejší (potřebujeme okamžité reporty), či zda musí probíhat velmi důkladné validace dat pro provedení procesů (jedná se např. o citlivá data). Po specifikaci těchto požadavků musíme zjistit, zda je můžeme skutečně otestovat a zda můžeme testovat automaticky. Důležité je také si ověřit, zda jsou specifikace testů a požadavků opravdu jasné a nemůže dojít ke špatnému výkladu nebo nepochopení z pozice designera či strany uživatele. Testování či hodnocení požadavků patří mezi běžné metody softwarového inženýrství a pro ETL procesy v něm nevznikají žádné odlišnosti.

- **Validace dat** - tato sekce je již oproti běžnému testování mírně odlišná. V rámci běžných testů také provádíme validaci dat - ovšem většinou jako součást nějakých jiných testů. V případě ETL procesů se dle mého názoru jedná o klíčovou část našeho testování. Data jsou pro datové sklady to nejpodstatnější a na našich procesech závisí, jakým způsobem bude vnímán výstup datového skladu. V této části je třeba hlavně kontrolovat, zda ze zdrojových dat odmítneme ta špatná, zda nahrazujeme nekorektně zadaná data či upravujeme a slučujeme důležité informace dle zadání, a když je možnost, tak i reportujeme špatná nebo problémová data. Musíme též ověřit, zda jsou data transformována korektně v rámci pravidel databáze. Dále také musí dodržovat business pravidla (nesmíme porušit informační hodnotu dat pro výsledné reporty). Dalším bodem je otestování unikítní hodnoty klíčových polí mezi datovým zdrojem a datovým skladem. Podstatnou a často opomíjenou částí je testování, zda nedošlo ke ztrátě dat při přenosu do datového skladu - tedy zda jsme v průběhu trasnformace či nahrávání nepřišli o část našich dat. Na závěr se doporučuje testovat hranice a možnosti naší databáze - abychom se v rámci ETL procesů nesnažili utvořit výsledek, který by cílová databáze datového skladu nebyla schopna přijmout.

[7] Tento výraz bych volně přeložil jako přijetí od běžného uživatele.

- **Integrační testy** - mezi integrační testy patří ověření sekvence a výsledků ETL dávkových úloh. V rámci úlohy probíhá mnoho činností, je tedy třeba ověřit nejen výsledek, ale i samotný průběh a jednotlivé sekvence v rámci jedné úlohy. Dále bychom měli ověřit prvotní nahrání záznamů do datového skladu, zda proběhlo úspěšně, datový sklad je dostupný a daří se nám záznamy nahrávat. Poté bychom měli ověřit průběžné nahrávání starších dat a porovnat s nově nahranými záznamy z datového skladu. Neměli bychom také opomenout testovat odmítnutí nevalidních záznamů našimi ETL procesy - díky kvalitně zpracovaným požadavkům můžeme snadno otestovat a vytvořit testovací data. Na závěr patří do integračních testů také testování generování chyb našich ETL procesů.

- **Report testy** - v této části testování se zaměříme na obsah našich dat. Důležité je provést kontrolu dat získaných z reportu a zdrojových dat - zda dostáváme validní záznamy a nedošlo v rámci transformací ke změně informace. V rámci skladu bychom měli také provést kontrolu SQL dotazů a ověřit tak rozdíly mezi zdrojovými a cílovými daty. Na závěr této části bychom měli ověřit zabezpečení dat a jejich případné šifrování či dodržení bezpečnostních pravidel.

- **User acceptance testy** - v testech user acceptance nás zajímá přijmutí dat ze strany zákazníka. Je tedy důležité si projít, zda naše výsledky odpovídají požadavkům. Zda byla dodržena business pravidla a je nutné získat od uživatele potvrzení, že systém ETL procesů generuje validní a správná data.

- **Výkonnostní testy** - kromě obsahu a procesu ETL procesů nás zajímá také jejich náročnost. Je třeba ověřit, zda nahrávání a procesování dat a dotazů se provádí v předpokládaných lhůtách. Měli bychom též ověřit, zda využíváme maximálních možností našich hardware a softwarových možností. Nesmíme zbytečně pouštět mnoho malých transformací a poté čekat na výsledky, když bychom byli schopni zpracovat mnohem větší dávku dat. Poté bychom si měli ověřit, jaké jsou samotné možnosti stroje, abychom byli schopni předvídat vývoj při nárůstu dat. Dále ověřit časovou a hardwarovou náročnost pro menší várku dat a poté pro větší a dle výsledků odhadnout, do jaké míry jsme schopni ještě rozšiřovat zatížení serveru či stroje, který se stará o ELT procesy.

- **Regresní testy** - tyto testy splňují stejné požadavky jako v běžné praxi. Jedná se tedy o využití běžných testů i při nové konfiguraci ETL procesů. Regresní testy by tedy měly být schopny testovat naše transformace nehledě (či pouze s malou obměnou) na změnu našich postupů.

Kapitola 3

Alternativy k ETL procesům

Pro naše potřeby a vzhledem k již ozkoušeným procesům jsme zvolili řešení transformací a nahrávání dat pomocí ETL procesů. Dostupné jsou však i jiné metodiky transformací dat - pokusím se tedy v krátkosti představit dvě alternativy k ETL procesům a nastínit jejich možné použití pro komplexnější uvedení do tématu datových transformací.

3.1 ELT

Jak je z názvu patrné, jedná se o **Extract-Load-Transform**. Rozvoj v tomto směru nastal hlavně v posledních pár letech. Důvodů je několik, v prvotních návrzích datových skladů se počítalo s oddělenou sekcí pro úpravu dat (data staging area či ODS, viz. předchozí kapitoly) a tomu bylo i uzpůsobeno provedení transformací ještě před nahráním dat do cílového systému. Dalším důvodem jsou rozvíjející se RDBSM systémy [8] - ty nám umožňují provádět databázové operace za mnohem kratší čas a přichází s dalšími funkcionalitami (nejčastěji zmiňovanou novinkou jsou Oracle Exadata). Díky tomu jsme nyní schopni provádět dříve příliš náročné či neproveditelné procesy přímo na databázovém stroji. V několika bodech shrnu pro a proti ELT procesů z článku společnosti Iri, která se zabývá datamanagementem [20].

3.1.1 Pro

- **ELT využívá RDBSM stroje** - není třeba utrácet peníze za další dodatečný hardware, který slouží pouze pro ETL nástroje. Tento bod však může být i záporem - v případě slabého stroje nám odebírá výkon pro běžné dotazování.

[8] Relational database management system - databázový management systém založený na relačním modelu.

- **ELT transformace jsou psány jazykem SQL** - opět se jedná o poměrně sporný bod. Pro databázového uživatele to může být snazší přístup než se učit s novým nástrojem, a tak můžeme ušetřit za školení či zaměstnání ETL specialisty. Zároveň je však otázkou, na kolik nám v současné době SQL poskytuje variabilitu oproti ETL nástrojům a jejich funkcím.

- **ELT transformace vykazují mnohem rychlejší datovou propustnost** - v rámci využití RDBSM stroje dosahují transformace mnohem větší datové propustnosti a tím i zrychlení. Zde ovšem můžeme polemizovat o ceně, která je za toto zrychlení několikanásobně vyšší.

3.1.2 Proti

- **Limitovanost v rámci dostupných ELT nástrojů** - v současné době je na trhu jen velmi málo ELT nástrojů a z těch hojně využívaných můžeme jmenovat pouze Oracle Data Integrator. Jsme tedy omezeni velmi vysokými pěněžními náklady a malou variabilitou výběru.

- **Ztráta detailního monitoringu** - se současnými možnostmi přicházíme o podrobné informace o procesech a hlavně o jednoduše zpracovatelná metadata. V rámci SQL je výrazně těžší sledovat průběh jednotlivých transformací, jejich výstup a zachycení všech podrobností nutných pro analýzu našich dat a našeho procesu.

- **Ztráta modularity na úkor nastaveného designu pro výkon** - vzhledem k zrychlení výkonu jsme limitováni v úpravách, které můžeme do trasnformací integrovat. Tímto tedy přicházíme o značnou flexibilitu, která se nám dostává v rámci ETL procesů.

- **Databázové úkony snižují výkon databázového stroje** - tento problém byl zmíněn již v bodech pro ELT procesy. Vzhledem k tomu, že příslušné transformace spouštíme na RDBSM stroji, ubíráme výkon na běžné dotazy či dotazy v rámci reportingu.

Obecně dle dostupných informací a dle mého názoru závisí na tom, jaké máme hardwarové a softwarové možnosti. ELT procesy oproti ETL procesům jsou mnohem závislější na použité architektuře a našich finančních zdrojích. Jejich klady jsem tak shrnul výše a některé jsem z článku vypustil, neboť byly natolik sporné, že se nedaly označit za přímé klady. Prozatím mi z mého úhlu pohledu připadá, že převažují zápory. Naproti tomu zde však máme zaběhlé a funkčí řešení Oracle Data Integration, které staví na bázi ELT procesů. Řekl bych, že je to způsobeno využitím kvalitních strojů, které jsou v rámci tohoto řešení od Oracle potřeba. Ačkoliv i Informatica v roce 2006 přidala možnost využití ELT procesů v rámci PowerCenter. Neumím si však toto řešení představit v rámci střední společnosti, která by musela obětovat velký

obnost peněz na toto řešení a nevím, zda by ji to v celkovém procesu přineslo dostatečnou protiváhu v kvalitě řešení transformací dat.

3.2 ETLT

V případě ETLT, **Extract, Transform, Load, Trasnform**, se jedná o kombinaci ETL a ELT řešení - tedy využití toho nejlepšího z obou předchozích. O tomto řešení se zatím zmiňují pouze některé knihy a poměrně málo článků. Přesto mi připadalo jako zajímavé a do budoucna využitelné. ETLT rozděluje transformace do dvou sekcí - rychlé, neblokující transformace závislé na častém výběru dat jsou provedeny předem, ještě před nahráním do skladu. Následně mohou být již určitá data použita pro reporting. Zbylá data, která ještě nejsou připravena pro předání cílovému uživateli, tak mohou být dále upravena na databázovém stroji a tím jsme tedy rozčlenili transformace do dvou nezávislých skupin.[21]. V rámci ETLT je velmi důležité si vybrat, které operace jsou vhodně provádět mimo databázový stroj a které provedeme až po nahrání. Vzhledem k nedostatku informací bohužel nejsem schopen poskynout bližší informace či klíčové vlastnosti výběru operací v rámci první a druhé sekce trasnformací. Vcelku mě překvapuje, že o tomto přístupu je prozatím velmi málo dostupných publikací a tento způsob zatím není nějak výrazněji rozvíjen či zhodnocen.

Pro nasazení tyto dva přístupy v rámci našeho prostředí nepřipadají v úvahu. Nevlastníme totiž dostatečně výkonný databázový stroj. V rámci záporů řešení ELT bychom se připravili o kvalitní přístup k metadatům a historizaci. Zároveň jsme limitování i nástroji, neboť jsem v rámci průzkumu nenarazil kromě placeného nástroje od společnosti Oracle na nějaké open source řešení umožňující tento přístup.

Část II

Praktická část

Kapitola 4

Analýza

V současné době je fakultní datový sklad postaven na dvou hlavních data martech. První data mart je zaměřen na výsledky studentů. Jeho zdrojovými systémy jsou KOS a Progtest. Druhý data mart je zaměřen na úvazky - zde nalezneme údaje o úvazcích zaměstnanců fakulty. Zdrojovým systémem je databáze úvazků.

4.1 Zdrojové systémy

- **Úvazky** - v tomto zdrojovém systému se nachází všechny potřebné informace v rámci pracovních úvazků zaměstnanců fakulty. Jelikož se jedná o fakultní nástroj, je pro správce fakultního datového skladu snazší získat potřebná data. K datům je také možno přistupovat častěji (dle potřeb aktualizace datového skladu) a data mart tak obsahuje aktuální data. Při zpracování se provede pouze záloha databáze a následně se na zálohovaných datech spustí potřebné ETL procesy. Upravená data jsou pak překlopena do data martu. Do budoucna bychom mohli uvažovat i o přímém napojení ETL serveru na tento zdrojový systém, neboť se nachází ve vlastnictví fakulty. Tento přístup by nám zajistil možnost aktualizovat data úplně nezávisle na zálohách databáze úvazků.

- **Progtest** - jedná se o fakultní testovací systém. Tento systém je určen pro kontrolu a odevzdání programovacích úkolů v rámci několika předmětů. Systém zaznamenává odevzdání úkolů studenty, počet neúspěšných odevzdání či například informaci, kdy student poprvé na úloze začal pracovat. Tyto informace se dají snadno použít na prognózu průběhu studia daného předmětu. Zároveň nám poskytují informace o obtížnosti jednotlivých úloh a srovnání napříč předchozími roky. Data z Progtestu jsou získávána požadavkem na správce - ten vygeneruje xml soubor. Opět se nad tímto souborem provedou potřebné transformace a data jsou následně nahrána do data martu studijní výsledky. Jedná se také

o fakultní nástroj - ovšem poněkud komplexnější, neboť neslouží pouze jako databáze výsledků, ale plní výše uvedené funkce (kontrolu úloh, opisování apod.). U tohoto zdroje můžeme počítat s kvalitním výstupem - formát i data jsou na velmi dobré úrovni. Objevuje se nám tu však zápor, a to závislost exportu na správci Progtestu.

- **KOS** - informační systém využívaný v rámci celé univerzity ČVUT. Obsahuje kompletní informace o studentech, předmětech a výsledcích studentů v daných předmětech. Jelikož se v současné době jedná o fakultní datový sklad, zajímají nás pouze data o studentech fakulty informatiky. Zde se data opět získávají žádostí na kontaktní osobu, která má umožněn přístup do KOSu. Jsme tedy vázáni na podání žádosti a následně čekání na vyhodnocení. Vzhledem k tomu, že jsem v rámci školních projektů měl možnost pracovat s anonymizovanými daty (pro potřebu reportů v rámci BI aplikace), dovolím si zhodnotit tento zdroj komplexněji. Data v něm jsou vzhledem ke stáří systému a jeho modifikacím poněkud nečistá. Často se setkáváme (hlavně u starších dat) s neúplností, špatně vyplněnými hodnotami sloupců a často i s nedodržením jednotné formy. V případě KOSu se tak jedná o nejobtížnější ETL transformace z těchto tří zdrojů. Do budoucna se však plánuje datový audit, který by měl data vyčistit a sjednotit. To by velmi usnadnilo integraci s datovým skladem. Zároveň byl vytvořen projekt KOSAPI, který nám již teď umožňuje získat určitá data z KOSu svépomocí. Bohužel jsme v tomto směru velmi limitování obsahem dat.

4.2 Forma provádění ETL procesů

Jak jsem nastínil v předchozí kapitole, všechny exporty jsou prováděny ručně. Na základě exportů používaných zdrojových systémů se provádí předpřipravené ETL procesy. Správce skladu vždy musí exporty překlopit na stroj, kde jsou zrovna prováděny, a poté svépomocí dané transformace spustit. Pro tyto potřeby je v současné době využíván nástroj Kettle od Pentaho. V rámci těchto transformací jsou poté sledovány výstupy a v případě úspěchu jsou upravená data překlopena do data martů. Toto řešení je funkční, jelikož jsou reporty prováděny nepravidelně a nárazově. Zároveň prozatím nejsou kladeny nároky na aktuálnost reportů a v případě nutnosti jsou data zaktualizována. Pro budoucí použití v rámci SSP [9] a reportování úspěšnosti zkouškových období bychom však potřebovali tento proces zautomatizovat. V současné době také tento proces nezahrnuje historizaci a zaznamenávání metadat.

[9] Spolupráce s průmyslem - projekt, který v rámci ČVUT zajišťuje propojení firem se studenty.

4.3 Cílové data marty

Prozatím má fakultní datový sklad dva cílové data marty. Těmi jsou data marty se studijními výsledky a úvazky. Jejich obsah jsem nastínil v rámci zdrojových systémů těchto dvou martů. V současné době se hlavně využívá data mart dw_studijnivysledky. Jsou z něj generována data pro SSP a občasné reporty ohledně úspěšnosti předmětů. Vzhledem k nově otestovaným možnostem se však očekává mnohem větší využití právě v rámci reportování průběžné situace v rámci předmětů a vyučujících. Také SSP bude do budoucna potřebovat, aby byl tento data mart v aktuálním stavu.

4.4 Vyhodnocení analýzy

Z výše uvedené současné situace vyplývá hlavní potřeba celý proces zautomatizovat. Cílem mé práce je ulehčení práce správce skladu a také zkvalitnění získaných informací. ETL server má za cíl nejenom transformace spouštět, ale také sledovat jejich průběh. Oproti současnému řešení bude zaznamenávat informace o průběhu a výsledcích do logů, bude rozšířen o historizaci a bude nasazeno sledování metadat. To vše nám v budoucnu poskytne možnosti zjistit, jak se data měnila v průběhu času a odkud jednotlivá data pochází. Také budeme moci sledovat, jakým procesem jednotlivé informace prošly a jak byly upraveny. Má práce zároveň umožní přestoupit na datový sklad dle Inmonova návrhu. V případě velkého centrálního skladu vznikne větší potřeba na řízení ETL transformací. V současné době je právě díky Kimballově návrhu značná volnost v nezávislosti jednotlivých data martů. Centrální sklad však bude potřeba udržovat aktuální nejlépe v rámci všech jeho zdrojových systémů. ETL server nám tak umožní naplánovat aktualizaci dat dle předem daného časového plánu. Vzhledem k náplni mé práce jsem sepsal pouze informační popis tvorby ETL procesů a dostupných systémů - podrobný popis lze nalézt v diplomové práci Stanislava Kuznetsova [7].

Kapitola 5

Příprava prostředí

Následující kapitola shrnuje instalaci potřebného operačního systému a samotného ETL nástroje Kettle. Jedná se o funkční minimum potřebné k běhu navrhovaného ETL serveru. Zároveň čtenáře seznámím s několika problémy, se kterými se může při nasazení jednotlivých systémů setkat.

- **Volba OS** - vzhledem k multiplatformnosti zvoleného ETL nástroje jsem nebyl limitován ve výběru operačního systému. Pro pilotní nasazení jsem zvolil GNU/Linux - konkrétně distribuci CentOS. Má volba Linuxového serveru se zakládala na nenáročnosti, udržitelnosti a open-source řešení. Vzhledem k maximálnímu využití hardwaru mi přijde právě toto Linuxové řešení nejšetrnější v rámci nároků. Samozřejmě vždy závisí na konkrétní distribuci a samotné hardwarové požadavky bychom mohli ještě snížit. Pro pilotní nasazení a testování je tato distribuce vhodná svou nenáročností. Zároveň je často využívána i v komerční sféře při poskytování např. virtuálních linuxových serverů. Udržitelnost vidím v snadné správě a udržování záloh - systém není problém kdykoliv zálohovat a poté obnovit bez větších obtíží. Oproti řešení založeném na operačním systému Windows zde vidím velkou časovou úsporu například právě v případné obnově systému. Posledním bodem je open-source řešení - díky volbě CentOS nejsme nuceni platit za poskytnutí licence na daný operační systém a zároveň je zajištěna podpora v následujících letech. I při případné změně distribuce nebude problém naše řešení převést.

- **Požadavky ETL nástroje** - před instalací Pentaho Kettle je nutné nainstalovat Java Runtime Environment verze 1.5 a vyšší [22], která je volně ke stažení. Já jsem konkrétně využíval verzi 1.7.75. Žádné další požadavky na instalaci ze strany Kettle nejsou. Tuto vlastnost považuji za velký klad - Kettle není závislý na mnoha doplňujících systémech a programech.

- **ETL nástroj** - pro nasazení jsem zvolil nástroj **Pentaho Kettle**. Důvody volby tohoto nasazení jsem popsal v předchozí kapitole 2.3. Nástroj je volně ke stažení na stránkách Pentaho. Při instalaci na linuxovém prostředí stačí nástroj pouze rozbalit ze staženého archivu a v případě nainstalované JRE je Kettle připraven k použití. Při spouštění může být poprvé pro uživatele matoucí, že se v rozbalené složce nenachází žádný soubor Kettle. Kettle je totiž propojením několika programů starajících se o ETL procesy. Najdeme zde programy Carte, Encr, Kitchen, Pan a Spoon. Program Carte slouží jako webový server pro vzdálené ovládání nástrojů Kettle. Další program Encr slouží pouze pro nastavení a volby šifrování hesla. Pro nás nejdůležitějšími nástroji jsou právě **Kitchen, Pan a Spoon**. Nástroj **Spoon** slouží pro návrh a implementaci ETL transformací - v něm můžeme jednotlivé transformace v přehledném prostředí navrhovat, sledovat jejich průběh a nastavit potřebné dodatečné parametry. **Pan** využijeme v případě, že chceme manuálně spouštět ETL transformace navržené v programu Spoon přes příkazovou řádku (například v rámci nějakého předpřipraveného skriptu). Má volitelné parametry typu vstupní soubor, logování a simuluje všechny možnosti nabízené programem Spoon. **Kitchen** je rozšířený Pan - oproti Panu umožňuje navíc spouštění úloh. Samozřejmostí je však i spouštění samostatných transformací. Opět nám nabízí mnoho volitelných parametrů. V případě volby například distribuce Windows by vás čekala právě jedna aplikace Kettle, která je kombinací všech předešlých programů.

- **Doplňující serverové nástroje** - pro náš server jsem nasadil ještě nástroj pro přístup do samotné databáze - **pgAdmin**. Také jsem doinstaloval **Pentaho BI server** - server sloužící pro reporty nad datovým skladem - umožňuje využití dalších Pentaho nástrojů například pro tvorbu interaktivních reportů.

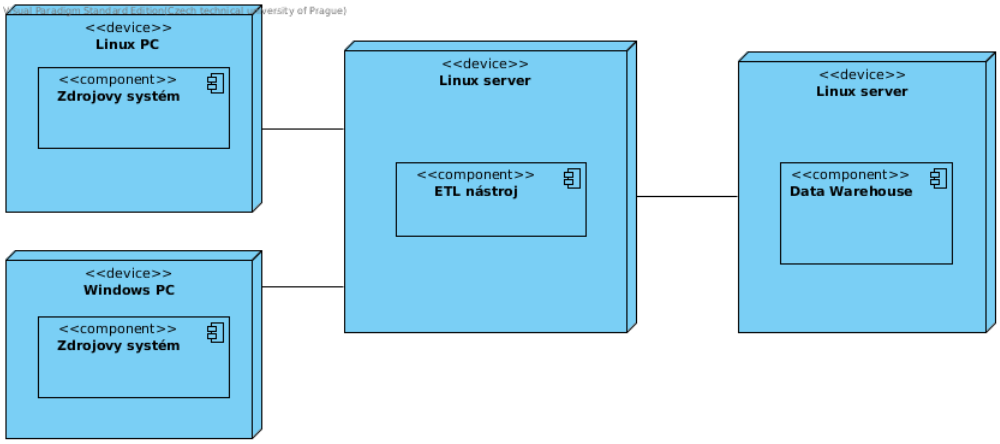
V závěrů bych rád ještě zmínil problém s nástrojem Spoon. Při spouštění program Spoon hlásil chybovou hlášku o špatném přístupu do paměti (jednalo se o java exception). Po pátrání v dokumentaci od společnosti Pentaha a navštívení jejich fóra jsem nalezl řešení v podobě úpravy konfiguračního souboru. Stačilo otevřít konfiguraci, běžně uloženou v souboru /.kettle/.spoonrc a následně do souboru připsat "ShowWelcomePageOnStartup=N". Vypnutí načítání uvítací stránky opraví tento záhadně vypadající problém. Rozhodl jsem se ho uvést, abych usnadnil administrátorovi hledání řešení v případě využití mé práce jako dokumentace.

Kapitola 6

Návrh

Pro potřeby ETL serveru je nejdůležitější pravidelné spouštění ETL úloh a zaznamenávání informací o proběhlých transformacích. Na obrázku níže jsem zachytil pomocí diagramu nasazení, jaké místo bude ETL server zastupovat v konkrétním výseku procesů v rámci datového skladu. Z libovolných zdrojových systémů budou proudit exporty na ETL server, kde budou zpracovány. Následně budou upravená data přesunuta do datového skladu. Diagram nezachycuje všechny komponenty, ale pouze komponenty přímo související s ETL serverem. Pro náš server jsem dále navrhl workflow [10], které popíši následně.

Obrázek 6.1: Diagram nasazení



[10] Pracovní či technologický postup, v podstatě sled navazujících událostí.

6.1 Předmluva k návrhu

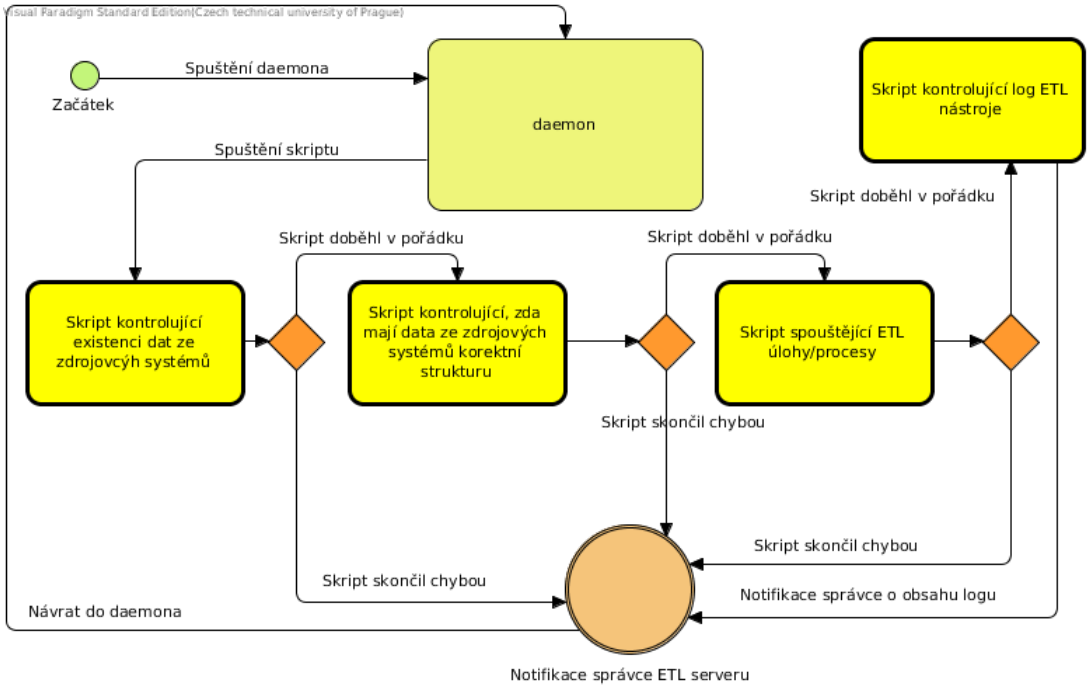
Jak jsem uvedl v předchozích kapitolách, v současné době jsou exporty dodávány správcem skladu. Do budoucna však určitě bude tento proces automatizován a reporty budou dodávany přímo ze zdrojových systémů na ETL server. Z tohoto důvodu jsem navrhl obecnější workflow, které primárně počítá právě s tímto způsobem. Při současném stavu je však také funkční a jedinou obměnou je ruční dodání dat ze zdrojových systémů. Toto workflow by však do budoucna mělo pokrýt všechna úskalí při automatickém procesu ETL transformací.

6.2 Workflow

Prvním krokem workflow, nebo-li pracovního či technologického postupu, je spuštění daemonu [11]. Následně proběhne kontrola, zda máme data ze zdrojových systémů. Když data chybí, celý proces končí, vrátíme se do stavu čekání a informujeme správce. V případě, že zdrojová data jsou dostupná - tak zkontrolujeme základní struktury souborů ze zdrojových systémů. Jestliže kontrola dopadne špatně, opět nepokračujeme a informujeme správce. Následně zavoláme ETL nástroj, aby provedl přednastavené transformace nad zdrojovými daty. Poté provedeme kontrolu logů z ETL nástroje a informujeme správce o výsledku. Workflow je zachyceno na následujícím obrázku 6.2 .

[11] Označení programu, který je spuštěn dlouhodobě a není v přímém kontaktu s uživatelem.

Obrázek 6.2: Návrh workflow běhu ETL transformací a jejich kontroly na ETL serveru



6.3 Souborová struktura

Pro usnadnění orientace ve zdrojových a logových souborech jsem navrhl následující souborovou strukturu na serveru. Tato struktura podporuje výše předepsanou workflow a dodává celému procesu zpřehlednění pro namátkové kontroly či řešení problémů.

- **src** - složka sloužící pro datové soubory ze zdrojových systémů. Jedná se o cílovou složku pro zdrojové systémy.
- **ok** - datové soubory jsou přesunuty do této složky po základní kontrole na konzistenci souborů ze zdrojových systémů. Jedná se o zdrojovou složku pro ETL nástroj.
- **archive** - do archivu jsou přesunuty soubory po provedení ETL transformací a jsou zde uchovány pro zpětnou kontrolu.
- **log** - ve složce log se tvoří soubory se záznamem informací ohledně proběhlých ETL úloh a transformací.
- **error** - složka error slouží k umístění souborů ze zdrojových systémů, které neprošly kontrolou.

Zároveň jsem připravil jednoduchý skript, který danou strukturu vytvoří - v případě nutnosti přesunu serveru či nasazení na jiný server.

6.4 Programová struktura

Pro řízení běhu ETL transformací vytvořím daemon, který bude pouštět jednotlivé skripty. Daemon bude logovat informace o spouštění a průběhu jednotlivých skriptů. Zároveň se dle výsledku spouštění daného skriptu rozhodne, zda pokračovat v procesu a nebo zda se jednalo o kritický bod a je třeba proces přerušit. Poté napíši skripty, které budou řídit jednotlivé kroky workflow. Bude se jednat o sadu čtyř skriptů. První a druhý skript se budou starat o data ze zdrojových systémů, ověří zda dorazila a zda mají správný formát. Dle výsledku kontroly budou data přesunuta na specifikované místo. Další skript bude spouštět samotné ETL úlohy a poté archivovat datové soubory. Poslední skript se postará o kontrolu log souboru, který bude generovat Kettle. Všechny skripty zároveň budou odesílat informační emaily správci v případě problémů či po úspěšném dokončení celé úlohy.

Kapitola 7

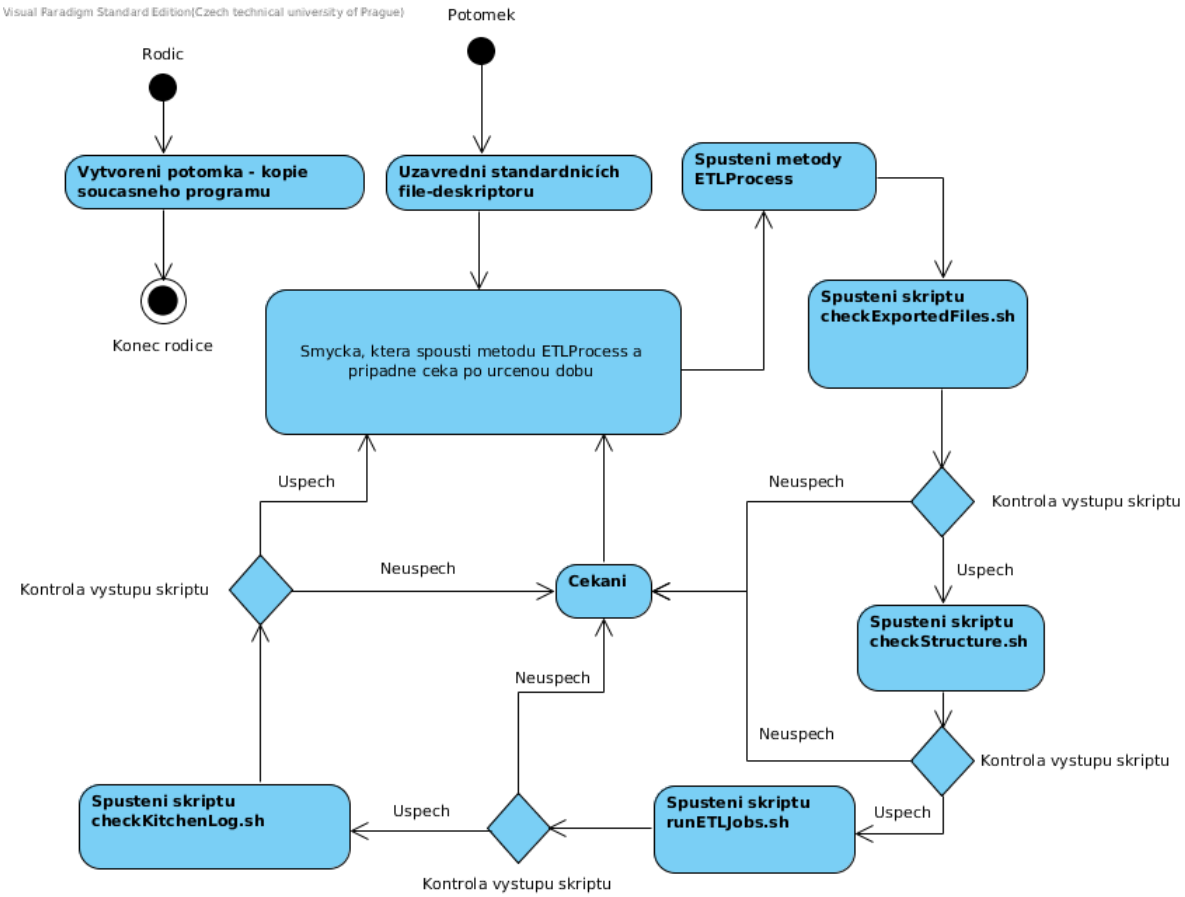
Implementace

7.1 Daemon

Daemon označuje program, který je spuštěn dlouhodobě a není v přímém kontaktu s uživatelem. Pro potřeby serveru jsem vytvořil daemon, který je psán v jazyce C++. Zvolil jsem tuto možnost, jelikož se jedná o efektivní řešení, které splňuje všechny požadavky, jež jsem od daemonu potřeboval, a zároveň je velmi snadné připravit dodatečné úpravy. Ve svém daemonu využívám knihovny syslog pro zapisování zpráv v rámci systémových logů. Následuje popis daemonu, který jsem doplnil o diagram pro lepší představu 7.1. Při spuštění si program vytvoří kopii - potomka pomocí metody fork [12]. Poté se původní program ukončí. Následně se v potomkovi uzavřou všechny standardní filedeskriptory. Poté už se spustí nekonečná smyčka, která vždy spustí metodu etlProcess. Metoda etlProcess se stará o spouštění jednotlivých skriptů Nejdříve spustí skript checkExportedFiles.sh a kontroluje jeho výstup. Při neúspěchu se metoda ukončí. Následně se spustí checkStructure.sh, tato procedura provede kontrolu struktury souborů ze zdrojových systémů. V případě chybového výstupu se zastaví celý další průběh a metoda se ukončí. Jestliže došlo k úspěchu, pokračuje se spuštěním skriptu runETLJobs.sh - tento skript zajistí spuštění ETL úloh. Opět kontrolujeme výstup tohoto skriptu a v případě neúspěchu ukončíme metodu. Když vše proběhne v pořádku, zavoláme metodu checkKitchenLog.sh, ta se stará o přezkoumání logu aplikace Kitchen nebo Pan či Spoon. V průběhu všech těchto procesů využívám knihovny syslog a všechny informace ukládám do logu umístěného dle systémového nastavení. Po ukončení metody využiji metodu sleep a uspím tento proces - zde si můžeme dle zadání navolit, po jakou dobu bude proces čekat na dodání nových dat ze zdrojových systémů. Kód můžete najít v příloze 10.1.

[12] Speciální systémové volání poskytované unixovým jádrem operačního systému. Jeho voláním vznikne z rodičovského procesu nový proces (potomek) tak, že se aktuální proces rozdvojí na dva identické.

Obrázek 7.1: Diagram aktivit daemou.



7.2 checkExportedFiles

První ze série shell skriptů. Tento skript projde složku src a ověří existenci datového souboru/datových souborů. V případě, že nalezne všechny soubory, vrátí návratovou hodnotu úspěchu a ukončí se. V případě, že některý ze souborů nenalezne, odešle jeho název společně s dodatečnými informacemi administrátorovi. Kód můžete najít v příloze 10.2.

7.3 checkStructure

Druhý skript se spustí v sekvenci, pouze pokud proběhl předchozí skript v pořádku. Zkontroluje hlavičky souborů ze zdrojových systémů a v případě, že je vše v pořádku, přenese soubory do složky ok. V opačném případě dojde k informování administrátora. Kód můžete najít v příloze 10.3.

7.4 runETLJobs

Tento skript dle názvu vezme soubory ze složky ok a spustí předpřipravené ETL úlohy pomocí programu Kitchen. Nejprve provede opět kontrolu zdrojových dat a souboru popisujícího ETL úlohy. V případě nutnosti ho můžeme snadno modifikovat, aby využil programu Pan. Při spouštění nastaví zvolenou úroveň logování informací průběhu transformací. V případě chyby dojde k notifikaci administrátora, ale v tomto kroku očekáváme spíše "chyby"v dodaných datech, které jsme schopni zachytit v rámci transformací. V předchozím kroku došlo ke kontrole struktury, a tím pádem by nemělo dojít k problému v rámci běhu ETL transformací. Po ukončení programu Kitchen se kontroluje návratová hodnota. Dle dokumentace programu Kitchen jsem sepsal jednotlivé chybové či úspěšné hlášky a ty odesílám správci. Po úspěšném proběhnutí ETL procesů se zdrojové soubory přesunou do složky archive. Kód můžete najít v příloze 10.4.

7.5 checkKitchenLog

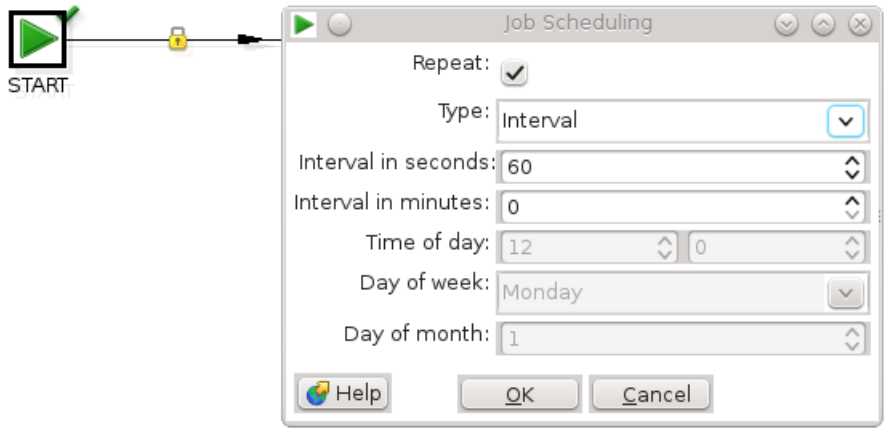
Poslední skript se stará o kontrolu logu programů ze sady Kettle. Projde zápis a dle předem dohodnutých pravidel bude kontrolovat důležitá hlášení. Můžeme zde navolit například konkrétní kroky v rámci ETL trasnformací či počet upravených řádků. Zde můžeme nastavit libovolný informační výpis, který by se administrátorovi mohl hodit k využití. Výsledek analýzy se odešle administrátorovi. Kód můžete najít v příloze 10.5.

7.6 Vestavěné spouštění v rámci programu Spoon

V závěru implemetace ještě zmíním možnost, kterou připravili vývojaři Pentaho pro plánované spouštění ETL úloh. Jedná se o vestavěnou funkcionalitu programu Spoon, která uživateli umožňuje plánované spouštění ETL úloh i přes grafické rozhraní. Na rozdíl od mnou implementovaného procesu však neumožňuje kontrolu zdrojových souborů a nezajišťuje další podružné úlohy. Uživatel může nastavit periody spouštění úloh a také úroveň logování informací. Tento postup uvádím pro případ, že by správce skladu měl zájem (například pro testování) nastavit pouze jednoduché plánování spouštění procesů. Nejdříve si otevřeme námi navrženou úlohu, následně místo klasického spuštění přes tlačíko run dvakrát poklepeme na položku Start (může mít i jiný název) a objeví se nám menu 7.2. Zde můžeme navolit libovolné opakování ETL úlohy či úloh. Poté se program Spoon postará o pravidelné spouštění a logování informací 7.3. Rozhodl jsem se toto spouštění uvést, neboť jsem ho na serveru také testoval v rámci zjištění funkcionality a možnosti sledování průběhu ETL úloh. Nabízí nám sice vcelku zajímavé průběžné informace, ale, jak bylo zmíněno

výše, je nutné, aby u programu byl přítomen např. administrátor či správce ETL procesů a celý proces kontroloval.

Obrázek 7.2: Menu umožňující nastavení plánování úloh.



Obrázek 7.3: Ukázka průběžného výstupu spouštění ETL úloh.

Kapitola 8

Testování

V kapitole zabývající se požadavky na ETL procesy 2.4 jsem zmínil několik testů, které by měly určit kvalitu procesu ETL transformací. Ty bych rád využil i v rámci testování funkcionality ETL serveru. Vzhledem k současnému stavu, kdy jsou ETL transformace již vytvořeny a běží dle určitých pravidel, nebudeme provádět testování požadavků. To probíhá před samotným návrhem ETL transformací. Ze sady testů jsem si vybral **integrační testy**. Ještě bych okrajově zmínil report testy, které provádí ve své práci Robert Kotlář [4]. Při historizaci si v rámci Pentaho BI serveru vytvořil několik vzorových grafických reportů pro získání informace o výsledku historizace. Do budoucna by nás také mohly zajímat výkonnostní testy, kdybychom uvažovali o změně procesního stroje. Tyto testy jsem se rozhodl nezařadit, neboť pro současné dávky dat se nejedná o příliš relevantní testy. ETL nástroj totiž současné transformace zpracuje za tak krátkou dobu, že nemůžeme příliš poměřovat náročnost.

Dále provedu kompletní otestování všech mnou navržených skriptů a běhu daemonu. Bude se jednat hlavně o otestování nestandardních datových vstupů, korektní odesílání zpráv správci skladu, kontrolu logů a samotný běh celé workflow.

8.1 Testování daemonu

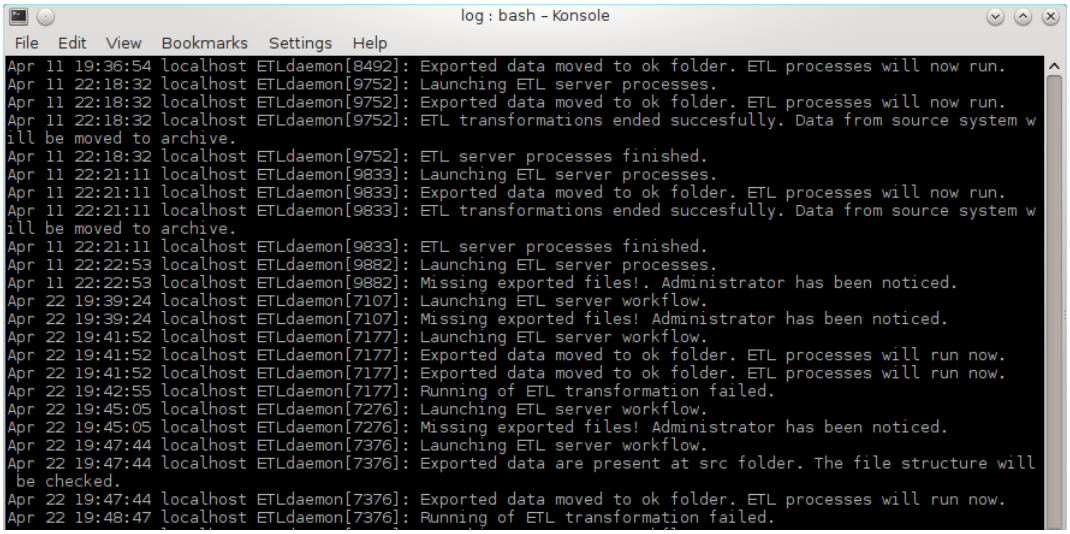
Při testování jsem připravil několik scénářů, které by mohly v průběhu spouštění ETL serveru nastat.

8.1.1 Chyba ETL procesu

Pro zobrazení výpisu zpráv logu daemonu jsem zvolil případ, kdy jsou dodána exportní data, mají korektní strukturu, ale objeví se chyba při běhu ETL transformací. Zde můžete také nahlédnout na strukturu výpisu chybových hlášek daemonu. Kompletní řešení daemonu naleznete v příloze 10.1. V logu vidíte detailní popis probíhajících úkonů. Nejprve vidíme zprávu o spuštění

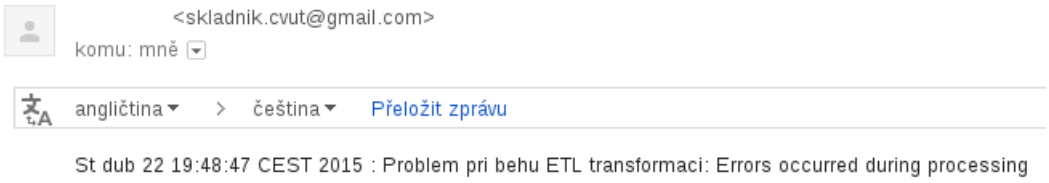
workflow ETL serveru. Poté můžeme vidět, že exportní data byla přítomna a přešlo se ke kontrole struktury. Ta proběhla také korektně, a tak byly spuštěny ETL skripty. Při nich došlo k chybě a ta byla korektně zalogována.

Obrázek 8.1: Ukázka logu daemona.



V návaznosti na tento problém probíhá kontrola výstupu spouštění programu Kitchen. Dle návratového kódu byl poté zaslán email s odpovídajícím problémem.

Obrázek 8.2: Chyba běhu ETL úlohy.

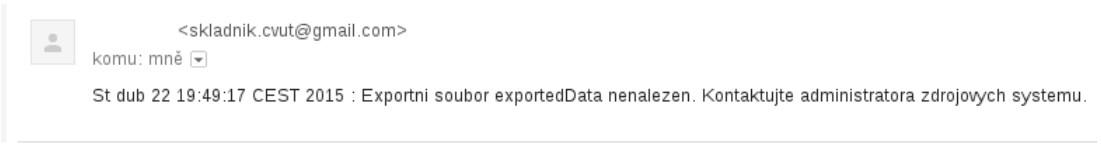


8.1.2 Chybějící zdrojové soubory či problémová struktura

Další scénář má dvě sekce.

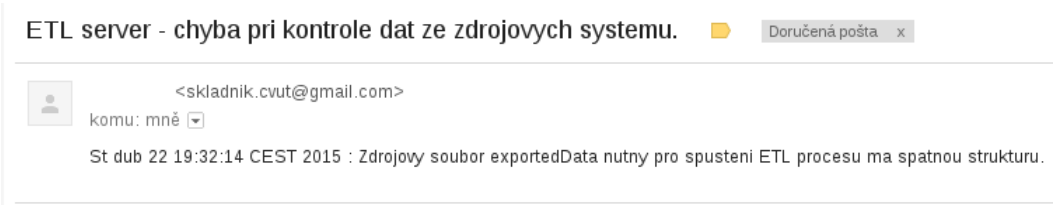
1. Nebyly nám dodány exportní soubory ze zdrojových systémů.

Obrázek 8.3: Chybějící soubor ze zdrojových systémů.



2. Byly nam dodány exportní soubory, ale mají špatnou strukturu.

Obrázek 8.4: Špatná struktura souboru ze zdrojových systémů.

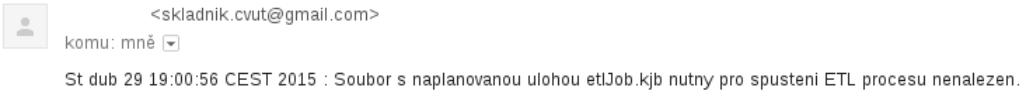


V obou případech daemon korektně přestal zpracovávat další kroky workflow a odeslal email správci ve znění uvedeném výše na obrázcích.

8.1.3 Chybějící soubor s ETL úlohami

V tomto případě jsem testoval případnou ztrátu definovaných ETL úloh. Daemon korektně odeslal email a ponechal zdrojové soubory ve složce ok, neboť by správce po tomto zjištění mohl jen doplnit soubor s definicí ETL úloh a tento skript spustit ručně. Nemusel by tak projít celý proces znovu.

Obrázek 8.5: Chybějící definice ETL úlohy.

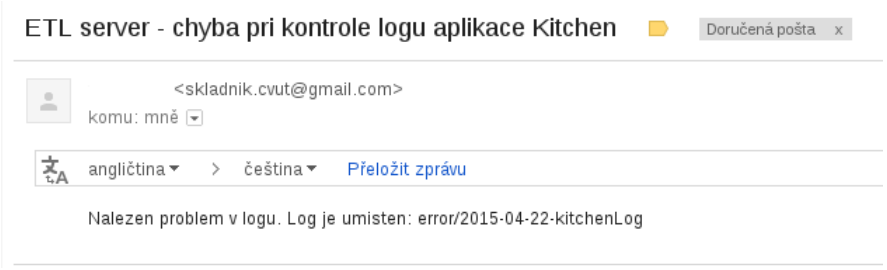


8.1.4 Chyba v datech - integrační test

V rámci integračních testů je třeba kontrolovat průchod jednotlivými body transformací. V tomto případě se jednalo o kontrolu, zda nebylo změněno uživatelské jméno studenta. Zde jsem opět využil napojení historizace mého

kolegy, která právě v rámci nahrávání změn do datového skladu kontroluje tento případ (jednalo se samozřejmě o testovací data). V rámci jeho procesu historizace je v ETL transformaci krok, který právě takové případy hlídá. Využil jsem tedy svého skriptu na parsování logu programu Kitchen. Ten zaznamenává, dle zvoleného nastavení, jaké řádky byly kde nahrány a kterými kroky ETL procesu prošly. Tuto změnu můj skript zachytil a log přesunul do složky error, následně informoval správce. Níže můžete vidět příklad struktury odeslaného emailu a ukázku logu programu Kitchen.

Obrázek 8.6: Chyba v dodaných datech - skript kontrolující log programu Kitchen.



Obrázek 8.7: Log aplikace Kitchen.

```
09:19:37 - nastala změna v parametru s SCD0 ?.0 - Sending row to false :nastala změna
09:19:37 - SCD 1 a 2.0 - Comparing id_student Integer(9) and id_student Integer(9) (c
09:19:37 - Odstraň pomocné sloupce.0 - Wrote row to next step: [781], [677], [677], [
09:19:37 - nastala změna v parametru s SCD0 ?.0 - Sending row to false :nastala změna
09:19:37 - SCD 1 a 2.0 - Comparing k_p_username String(8) and k_p_username String(8)
09:19:37 - SCD 1 a 2.0 - Comparing k_education_start Date and k_education_start Date
09:19:37 - Odstraň pomocné sloupce.0 - Got row from previous step: [3341], [2923], [2
09:19:37 - SCD 1 a 2.0 - Comparing k_study_program String(8) and k_study_program Stri
09:19:37 - nastala změna v parametru s SCD0 ?.0 - Sending row to true  :do nothing -
09:19:37 - Odstraň pomocné sloupce.0 - Wrote row to next step: [3341], [2923], [2923]
09:19:37 - nastala změna v parametru s SCD3 ? .0 - Sending row to false :Odstraň pomo
09:19:37 - SCD 1 a 2.0 - Comparing k_study_form String(8) and k_study_form String(8)
09:19:37 - SCD 1 a 2.0 - Comparing k_education_start_semester_code String(50) and k_e
09:19:37 - SCD 1 a 2.0 - Comparing k_education_end_semester_code String(50) and k_edu
```

Závěr

V rámci mé bakalářské práce se mi podařilo naimplementovat a otestovat ETL server. Dále jsem připravil pracovní workflow, díky kterému se celý proces spouštění ETL transformací zpřehlednil a automatizoval.

V první části práce jsem v rámci rešerše zjistil, že použití ETL nástroje není jedinou možností a nabízí se také vlastní naprogramování. To se však neukázalo jako vhodné pro použití v rámci fakultního či do budoucna celouniverzitního datového skladu. Poté jsem na základě rešerše zvolil vhodný ETL nástroj - konkrétně Pentaho Kettle, který nejvíce vyhovoval daným požadavkům. Dále jsem čtenáře seznámil s alternativami k ETL procesům, které by mohly být využitelné v případě potřeby optimalizace datových procesů na rozličném stroji pro ETL server. V závěru teoretické části jsem pro posouzení klíčových ukazatelů výkonnosti využil testování kvality a kontroly běhu ETL transformací.

V druhé části své práce jsem nejprve provedl analýzu současného řešení a na jejím základě navrhl strukturu ETL serveru. Pro pilotní nasazení jsem zvolil linuxový server a na něj nasadil zmiňovaný nástroj Pentaho Kettle. Pro automatizaci celého procesu jsem naprogramoval daemon napsaný v jazyce C++. Tento daemon se stará o celý proces a běh ETL transformací a také kontroluje log programu Kitchen (součást Kettle). Daemon mimo jiné loguje průběžné kroky spouštění celého pracovního workflow. V rámci své činnosti pak spravuje spouštění shell scriptů, které se starají o kontrolu dat ze zdrojových systémů, poté o spouštění samostatných ETL úloh pomocí nástroje Pentaho Kettle. V neposlední řadě také využívám vestavěného logování informací ohledně průběhu samotných ETL úloh a transformací a pomocí dalšího skriptu parsuji tento výstup. Na základě výsledku jednotlivých operací poté notifikuji správce datového skladu emailem, který obsahuje chybové hlášky a nebo notifikuji o úspěšném průběhu. V rámci otestování jsem poté navrhl několik scénářů, které prověřily robustnost systému a ozkoušely jeho reakci na běžné či nenadálé chyby. Pro testování jsem využil testovacích ETL transformací, které byly vylepšeny o pilotní nasazení historizace. Na těchto transfor-

macích jsem zkoušel funkcionalitu svého ETL serveru.

Do budoucna bude jistě po skutečném nasazení třeba upravit workflow ETL serveru dle delšího testování. Dalším rozšířením ETL serveru by bylo rozšíření ovládání celého procesu a jeho sledování například v rámci webové aplikace či webového rozhraní. To by velmi prospělo ke snadné orientaci a kontrole. Vzhledem k plánované přestavbě fakultního datového skladu je ETL server nutností, a proto se jistě ještě v budoucnu dočká vylepšení, která vyvstanou z potřeb nového skladu.

Literatura

[1] INMON, W. H.: ETL IN DW2.0 [online]. 2006, [cit. 2015-03-19]. Dostupné z: http://www.inmoncif.com/news/pdf/etlFN.pdf

[2] INMON, W. H.; STRAUSS, D.; NEUSHLOSS, G.: *DW 2.0 - Architecture for the Next Generation of Data Warehousing*. Elsevier Press, 2008.

[3] KIMBALL, R.; ROSS, M.: *The Data Warehouse Toolkit - The Definitive Guide to Dimensional Modeling, Third Edition*. Wiley Publishing, Inc, 2013.

[4] KOTLÁŘ, R.: *Historizace*. Diplomová práce, České vysoké učení technické v Praze, Fakulta informačních technologií, 2015.

[5] KREJČÍ, J.: *Metadata*. Diplomová práce, České vysoké učení technické v Praze, Fakulta informačních technologií, 2015.

[6] GEORGE, S.: ETL Tools Comparison [online]. 2015, [cit. 2015-03-15]. Dostupné z: http://searchbusinessintelligence.techtarget.in/tip/Inmon-vs-Kimball-Which-approach-is-suitable-for-your-data-warehouse

[7] KUZNETSOV, S.: *Datový sklad fakulty*. Diplomová práce, České vysoké učení technické v Praze, Fakulta informačních technologií, 2013.

[8] KIMBALL, R.; CASERTA, J.: *The Data Warehouse ETL Toolkit: Practical Techniques for Extracting, Cleaning, Conforming, and Delivering Data*. Wiley Publishing, Inc, 2004.

[9] NISSEN, G.: Is Hand-Coded ETL the Way to Go? Absolutely Yes, or Absolutely No, Depending... [online]. 2013, [cit. 2015-03-10]. Dostupné z: http://www.garynissen.com/etl-hand-code-or-tool/

[10] van der LINDEN, R.: ETL Tools Comparison [online]. 2014, [cit. 2015-03-15]. Dostupné z: http://www.etltool.com/etl-tools-comparison/

[11] INFORMATICA: PowerCenter Express - ETL Toolr [online]. 2015, [cit. 2015-03-18]. Dostupné z: https://community.informatica.com/solutions/pcexpress

[12] ORACLE: Take SOA Deployments to the Next Level with Oracle Data Integrator [online]. 2012, [cit. 2015-03-15]. Dostupné z: http://www.oracle.com/technetwork/middleware/data-integrator/learnmore/odi-for-soa-wp-1555852.pdf

[13] ORACLE: Oracle Technology Global Price List [online]. 2015, [cit. 2015-03-15]. Dostupné z: http://www.oracle.com/us/corporate/pricing/technology-price-list-070617.pdf

[14] ETL-TOOLS, T.: ETL Tools Comparison [online]. 2014, [cit. 2015-03-15]. Dostupné z: http://www.etltools.net/free-etl-tools.html

[15] BUTLER, M.: 5+ Free Open Source ETL Tools [online]. 2014, [cit. 2015-03-15]. Dostupné z: http://butleranalytics.com/5-free-open-source-etl-tools/

[16] REDICTIVEANALYTICSTODAY, T.: Top 15 Free Extract, Transform and Load, ETL Software [online]. 2014, [cit. 2015-03-15]. Dostupné z: http://www.predictiveanalyticstoday.com/top-free-extract-transform-load-etl-software/

[17] TALEND: Talend Open Studio for DI/DQ/BD Installation Guide [online]. 2015, [cit. 2015-03-18]. Dostupné z: http://www.talendforge.org/wiki/doku.php?id=doc:installation_guide

[18] CLOVERETL: CloverETL Community Edition [online]. 2007, [cit. 2015-03-18]. Dostupné z: http://doc.cloveretl.com/documentation/UserGuide/index.jsp?topic=/com.cloveretl.gui.docs/docs/cloverelt-community-edition.html

[19] PQATESTIN: ETL and Data Warehouse Testing [online]. 2012, [cit. 2015-03-18]. Dostupné z: http://www.pqatesting.com/our_ideas/blog/etl_and_data_warehouse_testing

[20] FRIEDLAND, D.: ETL vs ELT: We Posit, You Judge [online]. 2015, [cit. 2015-03-18]. Dostupné z: http://www.iri.com/blog/data-transformation2/etl-vs-elt-we-posit-you-judge/

[21] KOZIELSKI, S.; WREMBEL, R.: *New Trends in Data Warehousing and Data Analysis*. Springer, 2009.

[22] PENTAHO: Installing Kettle [online]. 2015, [cit. 2015-03-19]. Dostupné z: http://wiki.pentaho.com/display/EAI/01.+Installing+Kettle

Kapitola 9

Seznam použitých zkratek

ETL Extract trasnform load

XML Extensible markup language

BI Business intelligence

CSV Comma-separated values

ERP Enterprise resource planning

Kapitola 10

Příloha

Listing 10.1: Jednoduchý daemon pro řízené spouštění.

```
#include <sys/types.h>
#include <sys/stat.h>
#include <syslog.h>
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <errno.h>
#include <unistd.h>


#define DAEMON "daemon"

using namespace std;

/*
Metoda se stara o~beh shell skriptu.
Metoda dale kontroluje zda skripty probhely v~poradku
    a dle toho
se rozhoduje zda bude dale pokracovat
*/
void etlProcess(){
    syslog(LOG_NOTICE, "Launching␣ETL␣server␣workflow
       .");


    if(system("./checkExportedFiles.sh") == 0){
```

```
        syslog(LOG_NOTICE, "Exported␣data␣are␣present
            ␣at␣src␣folder.␣The␣file␣structure␣will␣be
            ␣checked.");
    }else{
        syslog(LOG_NOTICE, "Missing␣exported␣files!␣
            Administrator␣has␣been␣noticed.");
        return;
    }

    if(system("./checkStructure.sh") == 0){
        syslog(LOG_NOTICE, "Exported␣data␣moved␣to␣ok
            ␣folder.␣ETL␣processes␣will␣run␣now.");
    }else{
        syslog(LOG_NOTICE, "Missing␣exported␣files␣or
            ␣file␣structure␣wasn't␣correct!␣
            Administrator␣has␣been␣noticed.");
        return;
    }

    if(system("./runETLJobs.sh") == 0){
        syslog(LOG_NOTICE, "ETL␣transformations␣ended
            ␣successfully.␣Data␣from␣source␣system␣
            will␣be␣moved␣to␣archive.");
    }else{
        syslog(LOG_NOTICE, "Running␣of␣ETL␣
            transformation␣failed.");
        return;
    }

    system("checkKitchenLog.sh");
    syslog(LOG_NOTICE, "ETL␣server␣Workflow␣finished.
       ");

}

int main(int argc, char *argv[]) {
    setlogmask (LOG_UPTO (LOG_NOTICE));
    openlog ("ETLdaemon", LOG_CONS | LOG_PID |
       LOG_NDELAY, LOG_LOCAL0);


    pid_t pid, sid;

   //Vytvoreni noveho procesu.
```

```
    pid = fork();

    if (pid < 0) { exit(EXIT_FAILURE); }

    //Muzeme zavrit rodicovsky proces.
    if (pid > 0) { exit(EXIT_SUCCESS); }

    //Vytvoreni ID pro "child" proces.
    sid = setsid();
    if (sid < 0) { exit(EXIT_FAILURE); }

    // Uzavreni standartnich file deksriptoru.
    close(STDOUT_FILENO);
    close(STDERR_FILENO);
    close(STDIN_FILENO);

    while(true){
        etlProcess();
        sleep(<placetimehere>);
    }

    //Konec - uzavreme log.
    closelog ();
}
```

Listing 10.2: Skript kontrolujici dodana data ze zdrojovych systému.

```
#!/bin/bash
# Skript kontroluje, zda jsou dodana data ze
   zdrojovych systemu.

ok='ok';
archive='archive';
log='log';
error='error';
src='src';

dataExport='exportedData'

# Zde muze byt smycka pres vice souboru ze zdrojovych
    systemu.
# Pro nase potreby jsem v~ramci pilotniho nasazeni
   implementoval jeden konkretni export.
```

```
if [ -e ./"$src"/"$dataExport" ] ; then
        exit 0;
else
        echo "`date` : Exportni soubor $dataExport
           nenalezen. Kontaktujte administratora
           zdrojovych systemu." | mail -s "ETL server
           - chybejici export ze source systems" <
           admin_email>
        exit 1;

fi
```

Listing 10.3: Skript kontrolujici struturu dat ze zdrojovych systému.

```
#!/bin/bash
# Skript kontroluje, zda ma soubor ze zdrojovych
   systemu spravnou strukturu.
# V~pripade, ze je vse v~poradku, presune soubor/
   soubry do slozky ok.
# V~opacnem pripade je presune do slozky error.

ok='ok';
archive='archive';
log='log';
error='error';
src='src';

dataExport='exportedData'


# Doplnujici kontrola, zda soubory nebyly odstraneny.
if [ -e ./"$src"/"$dataExport" ] ; then
        pattern="Name,Surname,ID,Specialization,
           AverageMark"
        headline=`head -n 1 ./"$src"/"$dataExport" |
           awk '$1=$1'`
        if [ "$headline" == "$pattern" ] ;then
                mv ./"$src"/"$dataExport" ./"$ok"/"
                   $dataExport";
                exit 0;
        else
                echo "`date` : Zdrojovy soubor
                   $dataExport nutny pro spusteni ETL
```

```
                    procesu ma spatnou strukturu." |
                    mail -s "ETL server - chyba pri
                    kontrole dat ze zdrojovych systemu
                    ." <admin_email>
                mv ./"$src"/"$dataExport" ./"$error"/
                    "$dataExport";
                exit 1;
        fi
else
        echo "`date` : Zdrojovy soubor $dataExport
            nutny pro spusteni ETL procesu nenalezen."
             | mail -s "ETL server - chyba pri
            spousteni ETL trasnformaci" <admin_email>
        exit 1;
fi
```

Listing 10.4: Skript spoustejici ETL úlohy.

```
#!/bin/bash
# Scripts spusti ETL ulohu/ulohy pomoci programu
   Kitchen.

ok='ok';
archive='archive';
log='log';
todayDate="`date +%F`";
error='error';
src='src';
etlJobFile="etlJob.kjb"

dataExport='exportedData'

# Nejdrive zkontrolujeme, zda je ve slozce ok
   pritomen zdrojovy soubor.
if [ -e ./"$ok"/"$dataExport" ] ; then
        # Pote zkontrolujeme, zda je ve predem dane
           sloze umisten predpis ETL transformace/
           ulohy.
        if [ -e ./"$etlJobFile" ]; then

                /home/radimsky/Programy/data-
                    integration/kitchen.sh -file="
                    $etlJobFile" -level=Detailed -
```

```
    logfile="$log/$todayDate-
    kitchenLog";
retval=‘echo $?‘
echo $retval
# Chybove hlasky prevzaty z~
    dokumentace programu Kitchen -
    ponechal jsem je i v~anglictine
    pro lepsi nalezeni pri reseni
    problemu
if [ $retval = "0" ] ; then
        mv ./"$ok"/"$dataExport" ./"
            $archive"/"$dataExport";
        echo "‘date‘␣:␣ETL␣procesy␣
            dobehly␣v~poradku." | mail
             -s "ETL␣server␣-␣korektni
            ␣prubeh␣ETL␣procesu" <
            admin_email>
        exit 0;
elif [ $retval = "1" ] ; then
        mv ./"$ok"/"$dataExport" ./"
            $error"/"$dataExport";
        echo "‘date‘␣:␣Problem␣pri␣
            behu␣ETL␣transformaci:␣
            Errors␣occurred␣during␣
            processing" | mail -s "ETL
            ␣server␣-␣chyba␣pri␣behu␣
            ETL␣transformaci" <
            admin_email>
        exit 1;
elif [ $retval = "2" ] ; then
        mv ./"$ok"/"$dataExport" ./"
            $error"/"$dataExport";
        echo "‘date‘␣:␣Problem␣pri␣
            behu␣ETL␣transformaci:␣An␣
            unexpected␣error␣occurred␣
            during␣loading␣/␣running␣
            of␣the␣job" | mail -s "ETL
            ␣server␣-␣chyba␣pri␣behu␣
            ETL␣transformaci" <
            admin_email>
        exit 1;
elif [ $retval = "7" ] ; then
        mv ./"$ok"/"$dataExport" ./"
            $error"/"$dataExport";
```

```
                        echo "`date` : Problem pri
                           behu ETL transformaci: The
                           job couldn't be loaded
                           from XML or the Repository
                           " | mail -s "ETL server -
                           chyba pri behu ETL
                           transformaci" <admin_email
                           >
                        exit 1;
                elif [ $retval = "8" ] ; then
                        mv ./"$ok"/"$dataExport" ./"
                           $error"/"$dataExport";
                        echo "`date` : Problem pri
                           behu ETL transformaci:
                           Error loading steps or
                           plugins (error in loading
                           one of the plugins mostly)
                           " | mail -s "ETL server -
                           chyba pri behu ETL
                           transformaci" <admin_email
                           >
                        exit 1;
                elif [ $retval = "9" ] ; then
                        mv ./"$ok"/"$dataExport" ./"
                           $error"/"$dataExport";
                        echo "`date` : Problem pri
                           behu ETL transformaci:
                           Command line usage
                           printing" | mail -s "ETL
                           server - chyba pri behu
                           ETL transformaci" <
                           admin_email>
                        exit 1;
                fi
        else
                echo "`date` : Soubor s~naplanovanou
                   ulohou $etlJobFile nutny pro
                   spusteni ETL procesu nenalezen." |
                    mail -s "ETL server - chyba pri
                   spousteni ETL trasnformaci" <
                   admin_email>
                exit 1;
        fi
else
```

```
        echo "`date` : Zdrojovy soubor $dataExport
           nutny pro spusteni ETL procesu nenalezen."
            | mail -s "ETL server - chyba pri
           spousteni ETL trasnformaci" <admin_email>
        exit 1;
fi
```

Listing 10.5: Skript kontrolujici log aplikace Kitchen.

```
#!/bin/bash
# Skript zkontroluje log aplikace Kitchen pro
   konkretni specifikovane hlaseni.

ok='ok';
archive='archive';
log='log';
error='error';
src='src';

todayDate="`date +%F`";
kitchenLog="$todayDate-kitchenLog"
keyWord="nastala zmena v~parametru s~SCD0"
keyWord2="Sending row to true"

if [ -e ./"$log"/"$kitchenLog" ] ; then
        result="`cat ./"$log"/"$kitchenLog" | grep "
           $keyWord" | grep "$keyWord2"`"
        if [ -z "$result" ]; then
                mv ./"$log"/"$kitchenLog" ./"$archive
                   "/"$kitchenLog";
                exit 0;
        else
                mv ./"$log"/"$kitchenLog" ./"$error"/
                   "$kitchenLog";
                echo "Nalezen problem v~logu. Log je
                   umisten: $error/$kitchenLog" |
                   mail -s "ETL server - chyba pri
                   kontrole logu aplikace Kitchen" <
                   admin_email>
                exit 0;
        fi
else
```

```
        echo "`date` : Logovaci soubor $kitchenLog
           nenalezen. Zkontrolujte nastaveni
           spousteni ETL uloh." | mail -s "ETL server
           - chyba pri kontrole logu aplikace
           Kitchen" <admin_email>
        exit 1;
fi
```

Kapitola 11

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `src`
  - `impl` ..... zdrojové kódy implementace
    - `createFolderStructure.sh` ..... shell skript
    - `checkExportedFiles.sh` ..... shell skript
    - `checkKitchenLog.sh` ..... shell skript
    - `checkStructure.sh` ..... shell skript
    - `runETLJobs.sh` ..... shell skript
    - `daemon.cpp` ..... C++ skript
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `BP_Lenger_Radim_2015.pdf` ..... text práce ve formátu PDF