Návrh systému na automatickou tvorbu datového skladu

Martin Šárfy

23. září 2008

Abstrakt

Business intelligence nástroje se nástupem levných řešení stávají dostupné i pro malé a střední firmy. Výrobci SŘBD integrují jednoduché ETL a OLAP technologie přímo do svých produktů, k dispozici je i poměrně široká škála *open-source software* řešení. Jistou slabinou však pořád zůstává problematika tvorby samotného modelu datového skladu. Tento článek se snaží navrhnout systém na automatickou tvorbu modelu datového skladu na základě exportu dat z produkčních databází, včetně nástroje na jeho plnění. V prvním kroku jsou v datech rozeznány statisticky zajímavé údaje, na základě kterých je v dalším kroku navržen model datového skladu a transformační schema. Tuto definici pak může uživatel upravit a výslednou podobu předložit ETL stroji, který zajistí extrakci, transformaci a nahrání dat do datového skladu. V článku je takovýto *ETL engine* podrobně popsán, včetně návrhů na další rozšiřování. Celkově tak navržený systém umožňuje intuitivní a jednoduchou tvorbu datového skladu.

Klíčová slova: *ETL, ETL engine, data integration platform, schema mining, schema discovery, OLAP, data warehousing, business intelligence*.

Obsah

1 Úvod, motivace

Nejnákladnější částí zavádění *business intelligence* technologií je tradičně tvorba datového skladu ([S29], [S8], [S14]).

V posledních letech se tyto technologie stávají dostupnými i pro malé a střední firmy, a to díky integraci jednoduchých ETL a OLAP technologií výrobci SŘBD přímo do svých produktů buď zdarma nebo za nepatrně vyšší cenu, či díky dostupnosti použitelných *Open-Source* řešení.

Klíčový problém, kterému čelí společnosti při zavádění těchto technologií, je schopnost integrace dat ze svých produkčních systémů (v průzkumu [DIRR05] z listopadu 2005 přes 69 % respondentů tento problém označilo jako *velký* nebo *velmi velký*).

Ještě palčivější je tento problém u malých a středně velkých firem, které typicky nemají personál vyhrazený pouze na tento úkol. Požadavkem na integrační software je tedy maximální intuitivnost.

Nástroj, který teď představíme, se snaží být průvodcem uživatele při tvorbě datového skladu. Zapadá do moderního trendu nabízet software jako webovou službu, kterou v oblasti OLAP technologií v tuto chvíli nabízejí zejména následující společnosti:

- *SEA-Tab Software.* Produkt Pivot-Link, bez ETL nástroje.
- *Oco, Inc.* Komplexní BI software, pouze jednoduchý ETL nástroj.
- *GoodData, Corp.* Moderní webový OLAP nástroj, bez ETL části. Zatím pouze Beta verze.
- *LucidEra.* OLAP nástroj založený na OSS Pentaho Mondrian, bez ETL nástroje.

Tyto systémy předpokládají vytvoření vhodných struktur uživatelem použitím externího ETL nástroje. Námi navržený systém se pokusí zaplnit tuto mezeru a nabídne intuitivní způsob tvorby v případě jednoduchých datových skladů. Současně však nabídne i mechanizmus pro tvorbu složitých datových skladů, a to formou grafického modelování ETL transformací.

Základem pro samotný ETL *engine* nám bude teoretická práce *ARKTOS II* [SG02], dále *Open-Source* nástroj *Talend Open Studio* a funkční prototyp *The Bee Project* [MS03].

2 Architektura systému.

Software navrhujeme jako vzdáleně dostupnou službu (*software as a service*). V tomto prostředí lze pouze obtížně získat přímý přístup do produkční databáze (kvůli firemní politice, nastavení firewallů či proprietárním databázovým ovladačům), pragmatickým řešením však jsou pravidelné CSV exporty dat.

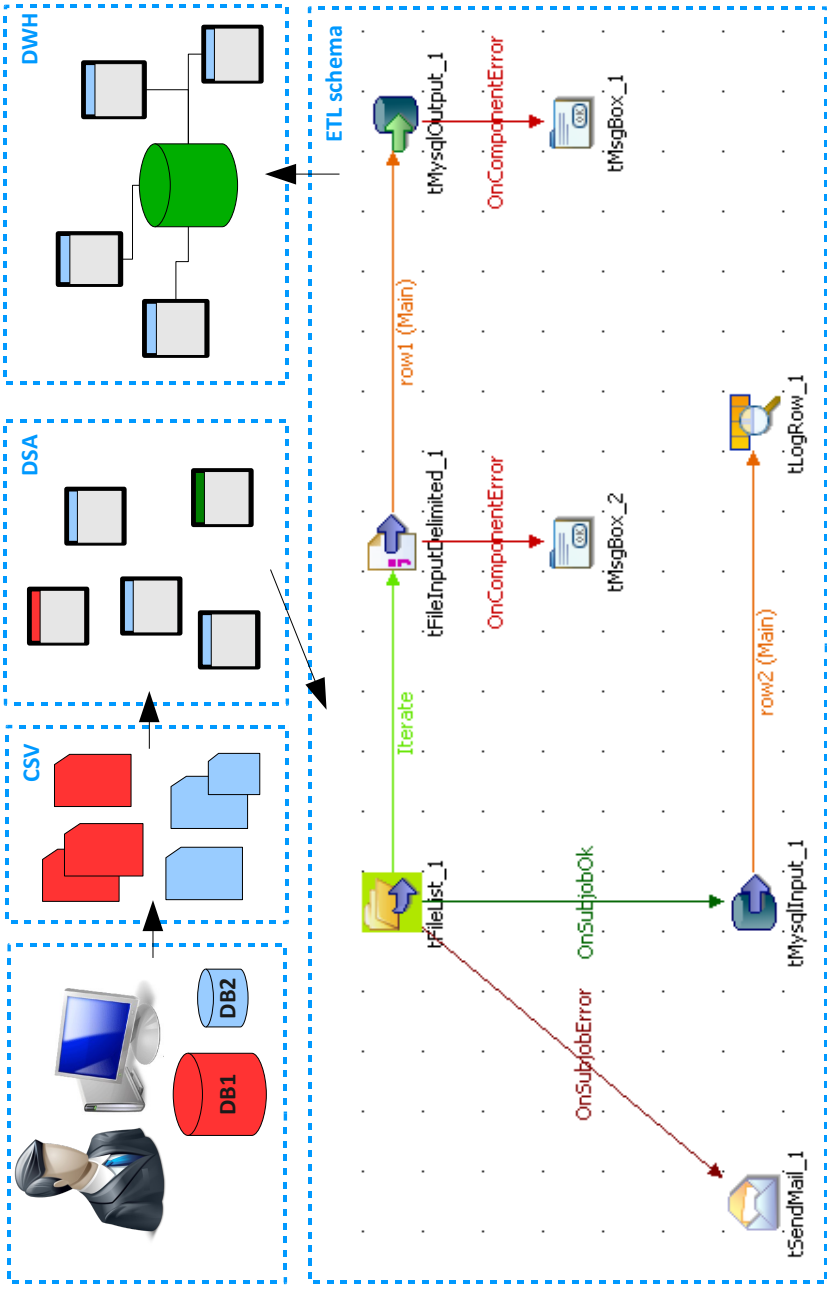
Námi navržený systém tedy předpokládá následující *work-flow*:

1. *Export dat z produkčních databází.* V prvním kroku uživatel zajistí CSV export dat z produkčních databází.
2. *Analýza exportovaných dat.* Systém v datech identifikuje datový model (tabulky, typy sloupců, primární a cizí klíče,..) a následně v něm pokusí rozeznat fakta, atributy, metriky, *look-up* tabulky, dimenze, apod.
3. *Předgenerování ETL schématu.* Na základě těchto údajů navrhne model datového skladu a ETL schéma (graf transformací) pro jeho (periodické) plnění.
4. *Úprava schématu.* V dalším kroku má uživatel možnost opravit chybné odhady systému případně doplnit další transformace, a to pomocí grafického nástroje pro editaci ETL schématu.
5. *ETL engine.* Takto upravené schéma je pak vstupem pro ETL engine, zajišťující (periodické) provedení příslušných transformací. Výstupem ETL enginu je naplněný datový sklad.

Implementace tohoto systému tedy spočívá v následujících částech:

1. Rekonstrukce databázového schematu.
2. Rozeznání OLAP prvků, vytvoření modelu datového skladu a tvorba příslušného ETL schematu.
3. Nástroj na vizuální opravu ETL schématu.
4. ETL engine.

Výsledný systém by měl být použitelný pro intuitivní tvorbu jednoduchých schémat, nicméně pořád dostatečně mocný (úplný) i pro náročnější případy.



Obrázek 1: Architektura navrženého systému.

3 Analýza exportu dat

Nejčastějším formátem pro přenos databázových dat je (v dnešní době už mírně zastaralý) formát CSV (*comma separated values*). Je podporován širokým spektrem výrobců, a to zejména pro jeho jednoduchou implementaci a nízkou režii.

Jeho zásadním problémem je nedostatek metadat, které s sebou nese. Naštěstí, jak si dál ukážeme, lze statisticky významné údaje s poměrně vysokou mírou úspěšnosti identifikovat automaticky.

Problém pochopení exportu dat rozdělíme na dvě fáze - na detekci relačního modelu produkční databáze a následně identifikaci prvků významných pro tvorbu datového skladu, čili rozeznání kandidátů na faktové tabulky, detekce sloupců obsahující metrické údaje, atributy, dále propojení výčtových hodnot na jejich *look-up* tabulky, detekci časových i jiných dimenzí či hierarchií dimenzí.

Veškeré odhady systému se dějí pod dohledem uživatele, který má za úkol schvalovat či opravovat rozhodnutí systému. Díky tomu chybný odhad systému *není* kritickým nedostatkem, konečné rozhodnutí zůstává v rukou uživatele.

3.1 Rekonstrukce datového modelu produkční databáze

Námi navržený algoritmus pro detekci databázového schématu ze vstupních dat má několik kroků ([DK08]):

1. Import vstupních dat z CSV souborů.
2. Detekce datových typů sloupců.
3. Nalezení primárních klíčů.
4. Nalezení cizích klíčů.
5. Export výsledného schématu.

Jak si ukážeme, žádný z těchto kroků nelze provést zcela automaticky, po každém je nutný (nebo alespoň vhodný) dialog s uživatelem pro potvrzení či upřesnění odhadu.

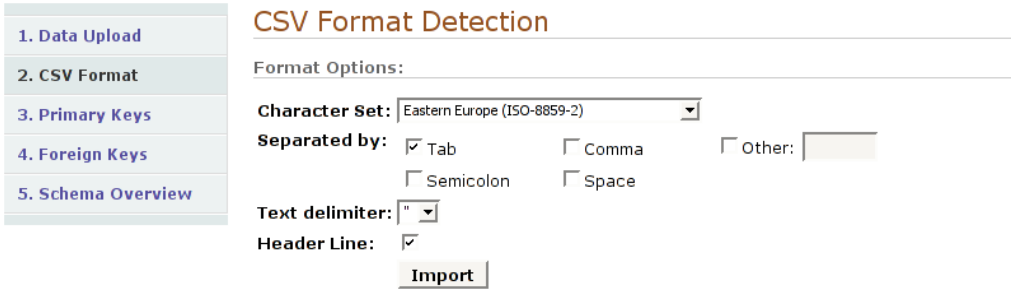
3.1.1 Import vstupních dat z CSV souborů.

Formát CSV nemá přesnou specifikaci. Ačkoliv jej doporučení RFC-4180 popisuje jednoznačně, jednotlivé implementace se mezi sebou

liší, a to zejména v následujícím:

- použitý znak pro oddělovač: kromě čárky se užívá tabulátor, středník, roura či jiný málo se vyskytující znak.
- textové hodnoty můžou ale nemusí být zapsány v uvozovkách[1].
- způsob zápisu zvláštních znaků: znaky můžou být zapsány v uvozovkách nebo použitím *escape*-sekvencí.
- použitá znaková sada a zápis znaku nového řádku (CRLF, LF, CR).
- u polí fixní délky zachovávání počátečních či ukončovacích mezer.
- nepovinný zápis názvů sloupců na prvním řádku tabulky.

Tyto způsoby zápisu lze algoritmicky odlišit s poměrně velkou mírou úspěšnosti (za předpokladu, že tabulka obsahuje alespoň několik řádků), schválení odhadu či jeho korekce uživatelem je však přesto důležitá.



Obrázek 2: Tradiční dialog pro import CSV souboru.

Pro určení znakové sady lze použít postupy běžně užívané např. ve webových prohlížečích [SK01].

Málokterá tabulka neobsahuje žádné číslo; pokud všechny hodnoty na prvním řádku začínají písmenem, resp. tvoří platný SQL identifikátor, jde o záhlaví tabulky a lze z něj vytáhnout i názvy sloupců. Jako počáteční jméno tabulky lze použít název souboru bez přípony.

[1]RFC-4180 doporučuje uvozovkovat alespoň textové hodnoty obsahující čárku, uvozovku nebo znak nového řádku.

3.1.2 Detekce datových typů sloupců

V tomto kroku určíme datové typy jednotlivých sloupců a taky porovnáme vyskytující se hodnoty se znalostní bází vzorů a pokusíme se určit i hlubší sémantiku některých sloupců. Příkladem vzorů znalostní báze můžou být kódy PSČ (ZIP), e-mailové adresy či telefonní čísla, což jsou vzory, které lze v datech lehce identifikovat.

Algoritmus projde všechny záznamy a pro každý sloupec vytvoří souhrnnou tabulku obsahující:

- počet celočíselných hodnot s úvodními nulami,
- počet celočíselných hodnot bez úvodních nul,
- histogram délky hodnoty ve znacích,
- histogram počtu slov (pro detekci kódů),
- počet prázdných hodnot,
- počet hodnot podobných nějaké formě zápisu data,
- počty hodnot vyhovující různým regulárním výrazům znalostní báze,
- počet unikátních hodnot ve sloupci.

Z těchto souhrnných údajů můžeme pak vyvodit následující závěry:

- SQL typ je jeden z: `INTEGER(N)`, `FLOAT(N,M)`, `CHAR(N)` či `VARCHAR(N)`, a to dle převládající vlastnosti. Maximální délku řetězce či počet platných cifer algoritmus určuje s rezervou.
- pokud histogram délek hodnot obsahuje pouze jednu hodnotu, má sloupec tuto fixní délku.
- *volitelně:* sloupec je `ENUM` pokud obsahuje několik málo různých nečíselných hodnot (shodné délky).
- *volitelně:* číselný sloupec je `NOT NULL` pokud neobsahuje prázdné hodnoty.

Jestliže významná většina záznamů vyhovuje některému regulárnímu výrazu ze znalostní báze, určíme podle něj i sémantický typ tohoto sloupce.

3.1.3 Nalezení primárních klíčů

Triviální algoritmus na nalezení jednoduchého primárního klíče spočívá v testování unikátních hodnot ve sloupci. Algoritmus je rovněž schopen hledat složené klíče testováním všech kombinací sloupců, zde však už narážíme na výpočetní náročnost takového algoritmu.

V praxi není obvyklé používat primární klíč složený z více než dvou sloupců, algoritmus je tedy omezen tak, že hledá klíče složené z nanejvýš dvou sloupců.

3.1.4 Nalezení cizích klíčů

Cizí klíče jsou základem pro modelování struktury objektů. Databázové schéma zná pouze jediný typ cizího klíče, který je používán pro modelování všech myslitelných typů struktur. Mezi běžně užívané typy vazeb mezi objekty patří:

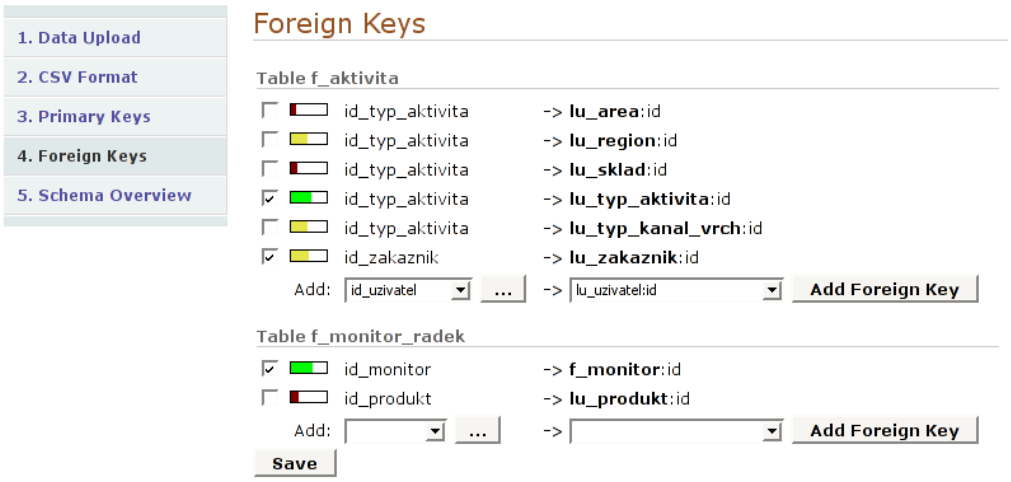
- objekt obsahující více podobjektů,
- objekty s volnou vazbou - číselníky, kódovníky
- struktura ukazující sama na sebe - hierarchie, stromy, lesy, grafy
- N:M vazby či vztahy přímo nesouvisejících objektů,
- zděděné třídy - odvozená tabulka obsahuje id a další atributy původního objektu.

Ve všech těchto případech odkazuje cizí klíč na nějaký primární klíč. Ačkoliv jde o jednoduché zjištění, není mi známá práce, která by tohoto faktu využívala.

Algoritmus pro hledání cizích klíčů tedy funguje následovně: Pro každý primární klíč (nalezený v předešlém kroku nebo zadaný uživatelem) složený z N sloupců algoritmus hledá ve všech tabulkách všechny kombinace sloupců řádu N. U každé kombinace vypočte následující charakteristiky pokrytí oboru hodnot primárního klíče (PK) a zkoumaného kandidáta na cizí klíč (FK):

- *platnost* - procento hodnot FK, které mají svůj vzor v PK.
- *pokrytí* - procento hodnot PK, které mají svůj obraz v množině FK.

Zatímco plnou shodu u obou parametrů lze očekávat pouze u prvního typu vazeb (tj. u objektů obsahujících více podobjektů), u ostatních musíme s požadavkem na procento pokrytí slevit.



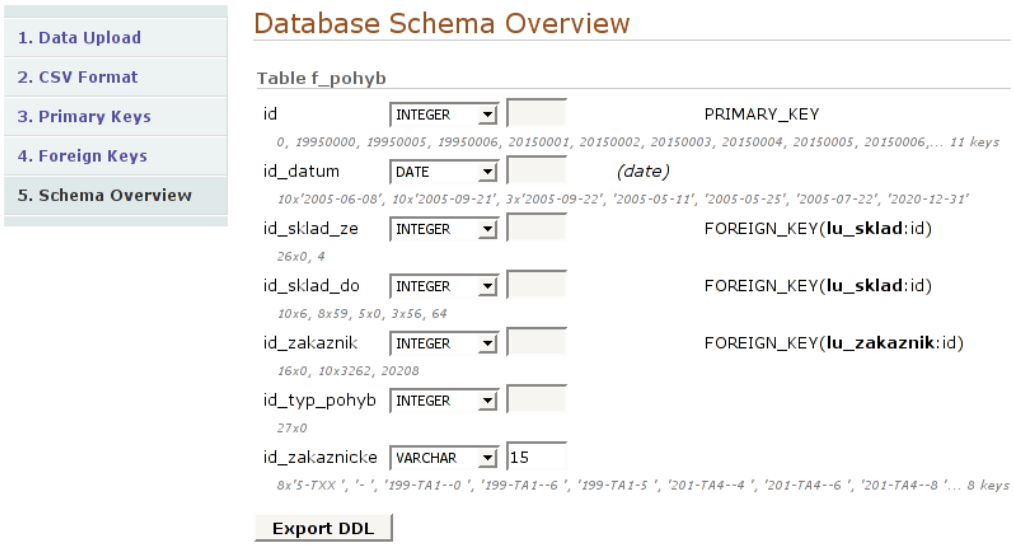
Obrázek 3: Detekce cizích klíčů, vyjádření pravděpodobnosti.

Na reálných datech je vidět, že pokud integritní omezení nehlídá samotná databáze (což je u produkčních systémů kupodivu celkem běžné), není dobré požadovat ani 100 % platných hodnot.

Jako vhodný kandidát na cizí klíč se empiricky zdají sloupce, které mají poměr platnosti nad 95 % a poměr pokrytí nad 20 %. Pokud sloupec odkazuje na více primárních klíčů, je jako správný vybrán ten, který obsahuje největší pokrytí.

Pokud databázové schéma obsahuje složené primární klíče, je potřeba hledat i složený cizí klíč. To se ukazuje jako výpočetně neúnosně složité, naštěstí se však složené primární klíče v praxi moc nepoužívají a náš obecný algoritmus hledání cizích klíčů tak většinou hledá pouze jednoduché klíče. Dále algoritmus při hledání cizího klíče bere do úvahy pouze sloupce, které mají datové typy v souladu s primárním klíčem, konkrétně buď jsou oba číselné nebo oba mají stejnou fixní délku.

Častou optimalizací databázového schématu je spojení několika sloupců fixní délky do jednoho. Z toho důvodu se (pokud má primární klíč fixní délku) dále testují i podřetězce sloupců, a sice začátek a konec kandidátního sloupce o délce odpovídající délce primárního klíče.



Obrázek 4: Zobrazení rozeznaného a opraveného schématu (zkráceno).

3.2 Určení schématu datového skladu

V tomto kroku se pokusíme určit některé charakteristiky datového modelu, které nám umožní vytvořit předpokládaný model datového skladu.

Datový model produkční databáze tvoří (orientovaný) graf tabulek pospojovaných cizími klíči. Některé tabulky obsahují fakta, stanou se tedy základem faktových tabulek v datovém skladě, kolem kterých se vytvoří hvězda či vločka tvořená tabulkami dimenzí z ostatních propojených tabulek.

Vhodným kandidátem pro faktovou tabulku jsou „dlouhé“ tabulky obsahující sloupec (či víc sloupců) s časovým údajem. Ten je vhodným indikátorem, že tabulka se v čase mění, a tedy bude pro uživatele zajímavá.

Uvedený postup je pouze nekompletním nástinem řešení problematiky, čekající na hlubší analýzu a prototypování. V tuto chvíli autor neumí celkovou úspěšnost algoritmu odhadnout, zdůrazněme však, že chybné odhady nejsou kritické.

Výstupem tohoto kroku by finálně měla být XML definice modelu datového skladu, včetně ETL schema pro jeho naplnění.

4 Definice ETL schématu.

Pro převod vstupních dat z produkční databáze do výsledného datového skladu je použito ETL schéma, jež je vstupem pro ETL stroj. Nyní si popišme, co vlastně ETL schématem a ETL strojem v našem kontextu myslíme.

ETL schéma nemá průmyslový standard, napříč literaturou neexistuje dokonce ani ustálená terminologie. Přesto se dá říct, že základní koncepce kolem ETL modelů je stejná - jde o orientovaný graf, kde uzly reprezentují transformace (akce, operace, komponenty, aktivity) a hrany toky dat (*data-flow*, řádky), případně specifikují hranice a pořadí jednotlivých úloh (*job*, *task*,. . .).

Pro naše potřeby nadefinujeme jednoduchý ale rozšiřitelný a široce použitelný ETL model. Tento model bude vycházet z prací *ARKTOS II* (dizertační práce [Sim04]), OSS systému *Talend Open-Studio* a funkčního prototypu [MS03], které však mírně rozšíří.

4.1 Transformace

Transformací rozumíme n-tici:

- *name* - jedinečné jméno transformace.
- *inputs* - seznam možných vstupních datových toků.
- *outputs* - seznam možných výstupních datových toků.
- *params* - parametry transformace.
- *flow* - model chování transformace (*main, input, output, filter, lookup*).
- *code* - třída v knihovně implementující kód transformace.
- *node* - uzel v síti, na kterém má být transformace spuštěna.

Transformace obecně může mít libovolný počet vstupních i výstupních datových toků, omezení závisí na hodnotě *flow*, podle kterého ETL stroj „orchestruje“ spouštění jednotlivých transformací:

- *input* transformace zajišťuje extrakci dat, nemá tedy žádný vstup, pouze generuje výstup (t.j. implementuje samotné čtení, např. transformace `MySQLSelectInput`).
- *output* transformace implementuje uložení vstupních dat, nemá tedy výstup, pouze vstup (např. transformace `ExcelFileOutput` či `ConsoleOutput`).

- *filter* transformace transformuje vstupní data na výstupní.
- *lookup* transformace vrací hodnotu asociovanou s klíčem.
- *main* transformace – o jejím chování vůči vstupům a výstupům nelze nic předpokládat, ETL stroj ji spouští v samostatném vlákně.

Použití síťového uzlu pro provedení transformace předpokládá instalaci potřebného softwarového vybavení a běh služby zajišťující nahrání a spuštění kódu. Implementačně i na údržbu nejjednodušší je na UNIX-ových uzlech využití systémové služby *Secure Shell* (*ssh*). Služba *ssh* bývá dostupná prakticky na všech strojích, navíc umožňuje zabezpečený přenos citlivých metadatových údajů (např. hesla do databáze apod.). Vzdálené volání je typicky využito pro přístup k lokálním datovým zdrojům, dále pro optimalizaci přenosu dat po síti či využití procesorového výkonu.

Knihovu transformací tvoří základní transformace obecného charakteru, nicméně lze ji také doplňovat uživatelským kódem.

4.2 Spojení transformací

Hrana spojující dvě transformace může být následujícího typu:

- *stream* - datový tok mezi dvěma transformacemi.
- *trigger* - umožňuje spustit cílovou transformaci při události (*OnDone* nebo *OnError*).
- *loop* - následující transformace je spouštěna opakovaně v cyklu (např. pro každý vstupní soubor,. . .), přičemž data může předat ve formě parametrů.

Spojení *trigger* a *loop* přináší možnost postupného, sekvenčního, spouštění transformací. Celé ETL schéma tedy dělí na jakési částečné *úkoly* (*job*, *task*), a to rozkladem grafu podle těchto spojení.

4.3 Spuštění ETL schématu

ETL stroj, který má za úkol vykonat samotné ETL, dostává na svém vstupu ETL schéma a provádí následující operace:

1. *Rozklad na úkoly.* V prvním kroku provede rozklad grafu na úkoly podle *trigger* a *loop* spojení. Vzniknuvší graf úkolů, pospojovaný

trigger a *loop* spojením, nyní dynamicky vyhodnocuje a spouští úkoly, které jsou schopny běhu, t.j. takové, které buď nemají *trigger* nebo nastala událost, která je spouští.

2. *Spuštění úkolu.* Pomocí síťové infrastruktury jsou obecně na různých uzlech vytvořeny objekty reprezentující transformace, které jsou následně propojeny *stream* spojeními datových toků. Pořadí inicializace je dáno orientací spojení, začíná se u transformací bez vstupů, pokračuje transformacemi, které mají všechny vstupy připojené, až do vytvoření všech transformací.

Po inicializaci všech transformací v úkolu je možné úkol spustit. Transformace běží navzájem paralelně, na více uzlech, ve více vláknech či procesech, čtou své vstupní datové toky, provádějí příslušnou operaci a výsledek posílají na výstupní toky. Nejsou-li během výpočtu na vstupním datovém toku připravena žádná data, je transformace operačním systémem pozastavena do doby, než se na datovém toku data objeví. Konec výpočtu je indikován uzavřením spojení a ukončením všech procesů transformace.

4.4 Knihovna standardních transformací

Další klíčovou vlastností dobrého ETL nástroje je široká paleta standardních transformací.

K základním transformacím patří podpora čtení a zápisu databázové tabulky, čili transformace:

- `SQLInput` - vykoná SQL dotaz, řádky posílá na výstup.
- `SQLOutput` - *zapíše*, *přepíše*, *přidá* nebo *smaže* řádek specifikovaný klíčem, při spouštění tabulku volitelně *vytvoří*, *vymaže* nebo *zahodí a znova vytvoří*.

Důležitou skupinou transformací jsou procesní transformace ovlivňující pořadí vykonávání dalších transformací a úkolů:

- `ForeachFileLoop` - vyhledá soubory určené maskou a pro každý zavolá následující transformaci.
- `IterateToFlow` - spustí následující transformaci opakovaně pro každý řádek vstupní tabulky.
- `Replicate` - duplikuje vstupní data, čímž umožňuje provést různé operace nad stejnými daty.

- `Unite` - spojuje data z více zdrojů na základě zadaného klíče (vhodné pro spojení dat z různých zdrojů).
- `Map` - mapuje data z jednoho či více zdrojů do jednoho či více výstupů, a to na základě mapovacího schematu.
- `Die` - ukončí provádění úkolu, s případným vypsáním zprávy.

Pro vytváření datových skladů jsou mimořádně potřebné transformace zajišťující databázový `JOIN` a *look-up* v tabulce:

- `Join` - nad zadanými sloupci tabulek provede zadaný typ `JOIN` dotazu, vrátí sloupce dle schématu.
- `Lookup` - pro klíče čtené na vstupu vrací hodnoty z *look-up* tabulky.

Tím výčet možných transformací samozřejmě nekončí, užitečné jsou i následující transformace:

- `Sort` - zadaným způsobem setřídí vstupní datový tok.
- `Aggregate` - u setřízeného vstupu agreguje sloupce dle zadané agregační funkce (`MIN`, `MAX`, `COUNT`,...).
- `Denormalize` - spojí hodnoty, u kterých se opakuje klíč.
- `SystemExec` - spustí systémový příkaz, výstup příkazu posílá jako datový tok.
- `ConsoleOutput` - posílaný vstup vypíše uživateli na obrazovku.
- `SendMail` - posílaný vstup odešle spolu se zadanou zprávou na e-mail.
- `HTTPFetch` - stáhne soubor ze zadané webové adresy.
- `MsgBox` - vypíše uživateli dialogové okno.

Produkční ETL nástroje zvyknou obsahovat stovky transformací (viz. Tab. 1), tyto však tvoří jakousi minimální množinu, která je dostatečná na většinu i náročnějších úkolů.

Uživatel samozřejmě může do knihovny doplnit vlastní transformace, přímo určené pro jeho specifické potřeby.

Business Intelligence	`[DB2|Ingres|Mondrian|MSSql|-MySQL|Oracle|Sybase]SCD`
Business	`[Centric|Salesforce|Sugar|Vtiger]-CRM[Input|Output]`
Custom Code	`Java`
Data Quality	`AddCRCRow, FuzzyMatch, IntervalMatch, ReplaceList, UniqRow, SchemaComplianceCheck`
Database	`[Access|AS400|DB2|Firebird|HSQLDb|Informix|Ingres|Interbase|JavaDB|JDBC|LDAP|MSSql|MSSqlBulk|MySQL|MySQLBulk|Oracle|Postgresql|SQLite|Sybase|Teradata]-[Input|Output|BulkInput|BulkOutput|Row|Commit|Connection]`
Database utilities	`CreateTable, ParseRecordSet`
ELT	`ELT[Oracle|Teradata|Mysql][Input|Map|Output]`
File	`File[Compare|Copy|Delete|Fetch|List], FileInput[Delimited|Positional|Regex|XML], FileOutput[Excel|LDIF|XML|Delimited], FileUnarchive, PivotOutputDelimited`
Internet	`FileFetch, FTP, Mom[Input|Output], SendMail, Socket[Input|Output], WebServiceInput, XMLRPC`
Logs & Errors	`Die, FlowMeter, FlowMeterCatcher, LogCatcher, LogRow, StatCatcher, Warn`
Misc group	`BufferOutput, ContextDump, ContextLoad, Loop, IterateToFlow, MsgBox, RowGenerator, Sleep, WaitForFile, WaitForSqlData`
Orchestration	`FileList, IterateToFlow, Loop, Replicate, Sleep, Unite, WaitForFile, WaitForSqlData`
Processing	`AggregateRow, AggregateSortedRow, Denormalize, ExternalSortRow, FilterColumn, FilterRow, Map, Normalize, Replace, Replicate, SortRow, Unite`
System	`RunJob, SSH, System`
XML	`AdvancedFileOutputXML, DTDValidator, File[Input|Output]XML, XSDValidator, XSLT`

Tabulka 1: Kompletní výčet transformací systému *Talend Open Studio*.

4.5 Možnosti optimalizace

Navrhnutý mechanizmus je implementačně dostatečně jednoduchý, nicméně pro reálné použití je vhodné integrovat i některé optimalizační prvky, které výrazným způsobem urychlí vykonání ETL schématu. Požadavek na rychlost je u ETL kritický, běžně je totiž potřeba zpracovat stovky tisíc či miliony záznamů během relativně malého časového okna.

Uvedeme zde proto několik výrazných optimalizací, které podstatným způsobem zkracují dobu exekuce ETL schématu:

1. *Využití* flow *charakteru transformace.* Jednoduchá implementace spouští všechny transformace v samostatných prováděcích tocích (vláknech či procesech), data jsou předávána pomocí meziprocesové komunikace (IPC) – síťovým spojením či UNIX sokety. Většina transformací je však z hlediska chování ke vstupům a výstupům jednoduchá a lze je převést na prosté volání funkce. Kupříkladu *input* transformace čtoucí z databáze může implementovat pouze funkci *read*, kterou bude následující transformace volat pro načtení dat. Stejný mechanizmus platí pro ukládání dat u *output* transformací. Rovněž z řady propojených filtrů lze poskládat řetězec volání transformačních funkcí. Zvláštní zřetel je však potřebá dát na transformace měnící počet záznamů mezi vstupem a výstupem, u kterých uvedený mechanizmus použít nelze.
2. *Použití UNIX soketů a sdílené paměti.* Jsou-li propojené transformace pouštěny na stejném uzlu, lze k přenosu dat použít místo TCP/IP spojení UNIX sokety, které oproti TCP/IP dosahují několikanásobně vyšších přenosových rychlostí. Další možností je implementace přenosu datového toku vytvořením sdílené paměti, fungující jako FIFO fronta mezi dvěma transformacemi, čímž se ušetří přesuny dat mezi adresním prostorem procesu a jádra operačního systému.
3. *Optimalizace čtení a ukládání dat do databázového stroje.* Standardní způsob pro čtení a zápis dat z a do databázové tabulky je dvojice SQL příkazů `SELECT` a `INSERT`. Databázové stroje však často poskytují i možnosti přímého čtení či zápisu, což je oproti obecným příkazům mnohem rychlejší. Příkladem může být příkaz `LOAD DATA INFILE` u MySQL databáze, který umožňuje nahrát data do tabulky z CSV souboru. Tímto souborem může být

rovněž UNIX soket, spojený s transformací. Kupříkladu taková `MySQLInsertOutput` transformace tedy může být optimalizována pro výrazné urychlení zápisu řádků, běží-li databáze na stejném uzlu a tento způsob ukládání dat je povolen.

4. *Volba uzlu transformace dle zátěže.* Díky distribuovanému prostředí, které navržená architektura využívá, je možné zvolit uzel, na kterém transformace poběží, dynamicky podle aktuální potřeby.
5. *SSH Mastering.* Tvorba jednoúčelových transformací má za následek, že i logicky jednoduché schéma obsahuje desítky až stovky transformací. Narozdíl od vlastních datových toků sice metadata potřebná pro správu takového množství vzdálených *ssh* procesů nejsou objemná, jsou však citlivá na dobu odezvy. Novější verze *ssh* protokolu umožňuje tzv. *ssh mastering*, sdílení jednoho navázaného spojení více procesy. To může výrazně urychlit dobu spouštění vzdálených transformací.
6. *Komprese a šifrování dat na vzdálených sítích.* Zatímco na dnešních vysokorychlostních lokálních sítích komprese dat zbytečně zatěžuje procesor, ve fázi extrakce dat z produkčních databází často narážíme na potřebu přenosu dat ze vzdálených poboček, které jsou připojené pomalým internetovým připojením. Komprese dat je v těchto případech potřebná, v tuto chvíli je rovněž vhodné využít vytvořenou síťovou infrastrukturu k přenosu sdíleného symetrického klíče k šifrování datového toku mezi pobočkami.
7. *Paralelní iterace.* U transformací generujících *loop* události na pořadí iterací často nezáleží (např. `ForeachFileLoop`) a cílový úkol je možné pustit paralelně v několika instancích. To může díky optimálnímu využití více procesorů či jiných zdrojů dramaticky zkrátit celkovou dobu provádění ETL schématu.

5 Závěr

Navrhovaný systém, bude-li implementován, se snaží zaplnit prázdné místo mezi nástroji pro podporu datové integrace. Návrh ETL stroje vychází z několika funkčních nástrojů, unikátní je režim vytváření modelu datového skladu uživatelským průvodcem.

Aktuální stav implementace pokrývá rekonstrukci databázového schematu z exportů dat, rovněž je hotov prototyp ETL stroje, který oproti běžným nástrojům umožňuje distribuovaný běh na více síťových uzlech.

Zbývající částí je algoritmus na rozeznání OLAP prvků v produkční databázi, vytvoření modelu datového skladu a tvorba příslušného ETL schématu. Schází také grafické prostředí pro tvorbu a úpravu ETL schématu.

Celek by však ve výsledku měl tvořit komplexní ETL prostředí s řadou unikátních prvků.

Použitá literatura

[RAL06] Rizzi, S., Abelló, A., Lechtenbörger, J., and Trujillo, J. 2006. *Research in data warehouse modeling and design: dead or alive?*. In Proceedings of the 9th ACM international Workshop on Data Warehousing and OLAP (Arlington, Virginia, USA, November 10 - 10, 2006). DOLAP '06. ACM, New York, NY, 3-10.

[S8] M. Demarest.: *The politics of data warehousing*. Dostupné z: *http://www.hevanet.com/demarest/marc/dwpol.html*

[TDWI03] Wayne Eckerson: *The Evolution of ETL. Evaluating ETL and Data Integration Platforms.* The Data Warehousing Institute (TDWI) 2003 Report Series.

[S14] B. Inmon.: *The Data Warehouse Budget*. DM Review Magazine, January 1997. Dostupné z: *http://www.dmreview.com/master.cfm?NavID=55&EdID=1315*

[S29] C. Shilakes, J. Tylman.: *Enterprise Information Portals. Enterprise Software Team.* Dostupné z: *http://www.sagemaker.com/company/downloads/eip/indepth.pdf*

[SG02] P. Vassiliadis, A. Simitsis, S. Skiadopoulos: *Modeling ETL activities as graphs*. In Proc. DMDW (Toronto, Canada, May 2002), pp. 52–61.

[MS03] Martin Šárfy: *Nástroj pro extrakci a transformaci dat.* Diplomová práce. Masarykova univerzita, Fakulta informatiky. 2003.

[Sim04] A. Simitsis.: *Modeling and Optimization of Extraction-Transformation-Loading (ETL) Processes in Data Warehouse Environments*. PhD Thesis, Athens, Greece, 2004.

[SK01] Shanjian Li, Katsuhiko Momoi: *A composite approach to language/encoding detection.* 19th International Unicode Conference, San Jose. 2001.

[DK08] Martin Šárfy: *Semi-automatická rekonstrukce databázového schématu na základě dat z tabulek.* To appear in proc.: Datakon 2008, Brno, Česká republika, 2008.

[DIRR05] Colin White: *Data Integration: Using ETL, EAI, and EII Tools to Create an Integrated Enterprise.* The Data Warehousing Institute (TDWI) 2005 Report Series. A 101communications Publications. November 2005.