TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Automatizované zpracování hydrogeologických dat

Bakalářská práce

Studijní program: B2646 – Informační technologie
Studijní obor: 1802R007 – Informační technologie

Autor práce: **Daniel Kendík**
Vedoucí práce: Mgr. Kamil Nešetřil, Ph.D.

Liberec 2017

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Automatized Processing of Hydrogeological Data

Bachelor thesis

Study programme: B2646 – Information technology
Study branch: 1802R007 – Information technology

Author: **Daniel Kendík**
Supervisor: Mgr. Kamil Nešetřil, Ph.D.

Liberec 2017

Technická univerzita v Liberci
Fakulta mechatroniky, informatiky a mezioborových studií
Akademický rok: **2016/2017**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Daniel Kendík**
Osobní číslo:	**M13000108**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Automatizované zpracování hydrogeologických dat**
Zadávající katedra:	**Ústav mechatroniky a technické informatiky**

Zásady pro vypracování:

1. Seznamte se s platformou Pentaho Business Intelligence (Data Integration, Reporting, Action Sequences) a datovým modelem HgIS.
2. S využitím platformy Pentaho vytvořte automatizovaný import dat, report vrtného profilu a automatizuje rozesílání reportů uživatelům.
3. Pro srovnání s použitým Pentaho Data Integration vytvořte import vybraných položek v jazyce Python.

Rozsah grafických prací: **dle potřeby dokumentace**

Rozsah pracovní zprávy: **30–40 stran**

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

[1] **BOUMAN, Roland L. a Jos van DONGEN, 2009. Pentaho solutions: Business intelligence and data warehousing with Pentaho and MySQL. Indianapolis: Wiley. ISBN 978-0-470-48432-6.**

[2] **CASTERS, Matt R., Roland BOUMAN a Jos van DONGEN, 2010. Pentaho Kettle solutions: building open source ETL solutions with Pentaho Data Integration. 1st ed. Indianapolis: Wiley. ISBN 978-0-470-63517-9.**

[3] **MATTÍO, Mariano García a Dario R. BERNABEU, 2013. Pentaho 5.0 Reporting by example: Beginner?s guide. Birmingham, UK: Packt. ISBN 978-1-78216-225-4.**

[4] **SUMMERFIELD, Mark, 2010. Python 3: výukový kurz. Přel. Lukáš KREJČÍ. Brno: Computer Press. ISBN 978-80-251-2737-7.**

Vedoucí bakalářské práce: **Mgr. Kamil Nešetřil, Ph.D.**
Ústav mechatroniky a technické informatiky

Datum zadání bakalářské práce: **10. října 2016**

Termín odevzdání bakalářské práce: **15. května 2017**

L.S.

prof. Ing. Zdeněk Plíva, Ph.D.
děkan

doc. Ing. Milan Kolář, CSc.
vedoucí ústavu

V Liberci dne 10. října 2016

Prohlášení

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

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu TUL. Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

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

Současně čestně prohlašuji, že tištěná verze práce se shoduje s elektronickou verzí, vloženou do IS STAG.

Datum: 15.5.2017

Podpis:

Poděkování

Rád bych poděkoval vedoucímu práce, doktorovi Kamilu Nešetřilovi, za jeho vstřícný přístup, ochotu a podporu při konzultacích a realizaci této práce.

Abstrakt

Tato práce popisuje postup při tvorbě návrhu reportu a transformací, které zajistí zpracování dat o hydrogeologických vrtech z formátu XML a v požadovaném formátu tato data zapíše do SQL databáze. Následně dojde k vytvoření a zaslání reportu uživateli prostřednictvím emailem. První část práce popisuje použité prostředky, zdrojová data a výstupní formát. Následuje popis samotné realizace transformací dat, které byly vytvořeny v nástroji Pentaho Data Integration. Dále popisuji tvorbu definice reportu vytvořeného v Pentaho Report Designer a také vlastní implementaci převedení dat z XML souborů do výstupních CSV souborů pro zápis do databáze pomocí programovacího jazyka Python. Nakonec je zhodnoceno použití zvolených nástrojů, jejich přednosti a nedostatky, a zejména je srovnávána tvorba transformací dat pomocí Pentaho Data Integration s vlastní implementací v Pythonu.

Klíčová slova

datové transformace, ETL, Kettle, Pentaho Data Integration, Pentaho Report Designer, XML, Python

Abstract

This work describes the procedure of making transformations that will ensure processing data about hydrogeological wells from the XML format and insert them into SQL database in required format. Subsequently, will be create a report and send to the user via an email. The first part of work describes the used resources, the source data and the output format. This is followed by the description of data transformations implementation that was made in Pentaho Data Integration tool, definition of report created in Pentaho Report Designer and also description of my own implementation of data transformation from XML files to CSV output format for inserting to database by programming language Python. In the last part of work is evaluation of used tool, its advantages and weaknesses. In particular, it compares the creation of a transformation in Pentaho with my own Python implementation.

Keywords

data transformation, ETL, Kettle, Pentaho Data Integration, Pentaho Report Designer, XML, Python

Obsah dokumentu

Seznam tabulek a obrázků

Tabulky

Obrázky

Seznam zkratek a termínů

Tabulka 1: Seznam zkratek a termínů

ETL	Zkratka Extract, Transform, Load – datové úlohy čtení, transformace a nahrávání dat.[1]
PDI	Pentaho Data Integration, neboli Kettle je produkt společnosti Pentaho poskytující mechanismy ETL, tedy načtení, transformace a nahrávání dat. [1]
XML	Extensible Markup Language je obecný značkovací jazyk určený především pro výměnu dat mezi aplikacemi a pro publikování dokumentů, u kterých popisuje strukturu z hlediska věcného obsahu jednotlivých částí
Komponenta	Komponenta je předdefinovaný prvek v grafickém prostředí Kettlu zvaném Spoon, který je součástí transformace.
Transformace	Transformace je množina komponent, která nějakým způsobem zpracovává data. Obvykle je součástí „jobu", ve kterém provádí pouze část celkového zpracování dat.
Job	Job je označení pro celkové zpracování vstupních dat do požadované výstupní podoby. Obvykle se skládá z několika transformací, které provádí větší části celkového zpracování dat.
Element	Element je párový tag (<tag></tag> nebo <tag />) který má svou hodnotu uloženou mezi počátečním tagem a koncovým tagem (<tag>hodnota</tag>) nebo jako atribut (<tag atribut=hodnota>).
XPath	Počítačový jazyk, který slouží k adresování jednotlivých elementů v XML souboru.
Report	Výstupní zobrazení zpracovaných dat, obvykle je ve formátu PDF, nebo HTML.

Úvod

Hlavním cílem této práce je vytvořit ETL transformaci v prostředí Pentaho Data Integration (Kettle), která bude zpracovávat data o hydrogeologických vrtech a jimi zkoumané hornině, dále vytvořit definici reportu v Pentaho Report Designer, který tato data bude přehledně zobrazovat a následně zajistit automatické rozesílání těchto reportů uživatelům pomoci emailu.

Datová transformace má zajistit načtení dat ze vstupních XML souborů a upravit je do požadovaného formátu. Následně uložit takto zpracovaná data do tabulek v databázi, která slouží jako výměnný formát pro další zpracování dat.

Zpracovávaná data pochází z aplikace Geologicky dokumentované objekty (GDO), provozované Českou geologickou službou (ČGS). Vstupní data jsou ve výměnném formátu XML podle mezinárodního standartu projektu eEarth. Obsahují základní informace o geologických vrtech (např. název, souřadnice, rok vzniku, hloubka, účel, atd.) a litologická data (složení horniny v jednotlivých vrstvách vrtu).

Výstupní formát dat je předem definovaná databázová struktura, která je používána jako výměnný formát pro další zpracování dat. Byla vyvinuta na TUL v rámci projektů a vychází z formátu programu EnviroInsite. Obsahuje tabulky pro uložení různých typů dat, v této práci byly využity pouze tabulky „Wells“ a „Borings“. Do tabulky „Wells“ se ukládají základní informace o vrtu a do tabulky „Borings“ se následně ukládají záznamy o složení horniny v jednotlivých hloubkách vrtu.

Dalším cílem práce je porovnat implementaci zpracování dat v PDI s vlastním řešením pomocí programovacího jazyka Python. Úkolem je zhodnotit výhody i nevýhody obou postupů, složitost jejich implementace a také náročnost modifikace pro ostatní uživatele.

1. Seznámení s problematikou

1.1. Vstupní data

Vstupní data pochází z archivu Geofond České geologické služby, která se zabývá sběrem a zpracováním údajů o geologickém složení území České republiky. Data jsou v podobě XML souborů. Každý XML soubor obsahuje informace o samotném vrtu (např.: název vrtu, klíč GDO, typ vrtu, účel vrtu, hloubka, atd.), dále obsahuje data o zkoumané hornině v jednotlivých vrstvách vrtu, kde je každá vrstva popsána hloubkou ve které se nachází, typem horniny a jejími vlastnostmi (viz obrázek 1).

```
<?xml version="1.0" encoding="UTF-8"?>
<borehole id="492719">
    <country>Česká republika</country>
    <language>česky</language>
    <nameDatabase>GDO</nameDatabase>
    <nameBorehole>S-2</nameBorehole>
    <locationMethod>odečteno z mapy</locationMethod>
    <ObjectType>vrt svislý</ObjectType>
    ...
    //Další data o vrtu
    ...
    <intervalseries>
        <gml:featureMember>
        <interval>
            <depthTop>0</depthTop>
            <stratigraphy1 value="Kvartér" code="Q"/>
            <rockName value="navážka" code="NVZ" rank="1"> </rockName>
        </interval>
        </gml:featureMember>
        <gml:featureMember>
        <interval>
            <depthTop>0.20</depthTop>
            <stratigraphy1 value="Kvartér" code="Q"/>
            <rockName value="hlína" code="HLN" rank="1">
            <colour value=" žlutá hnědá" code="Y H"/>
            <qualifier value="prachový" code="PRY"/>
            <qualifier value="vlhký" code="VLH"/>
            <qualifier value="měkký" code="MKK"/>
            </rockName>
        </interval>
        </gml:featureMember>
        ...
        //Další intervaly
        ...
    </intervalseries>
</borehole>
```

Obrázek 1 - Vstupní XML data

1.2. Datový model HgIS

Hydrogeologický informační systém (HgIS) spravuje hydrogeologická data, vytváří z nich různé výstupy (např.: grafy, tabulky, mapy, geologické mapy, atd.), které zprostředkovává. Pomocí ETL nástrojů zpracovává data z mnoha zdrojových formátů a ukládá je do datového skladu, kterým je databáze PostgreSQL. [2]

Datový model HgIS vychází z databáze programu EnviroInsite (viz tabulka 2). Datový model EnviroInsite definuje pole, která je třeba zadat pro zobrazení v EnviroInsite. Pro uložení všech dat, s nimiž HgIS pracuje, byl datový model EnviroInsite rozšířen tak, aby umožnil uložení popisných dat o objektech (vrty, studny), podmínkách vzorkování podzemní vody a časových intervalech. Dále, aby obsahoval zejména číselníky, kódovníky a pomocná data pro načítání dat do systému, pro převod jednotek a veličin a pro přejmenování. [2]

Tabulka 2 - Tabulky původního datového modelu EnviroInsite [2]

Tabulka	Popis
Wells	Identifikace objektů (zejm. vrtů, studní) – jejich souřadnice a další údaje
Borings	Popis geologických vrstev
Stratigraphy	Vymezení geologických vrstev a hydrostratigrafických jednotek pro vytvoření řezů a 3D geologických modelů
Well Construction	Výstroj studny (např. plná a perforovaná pažnice)
Fill	Obsyp a těsnění vrtu
Screens	Vzorkovaný hloubkový interval (otevřený úsek vrtu či hloubka odběru zeminy)
Constituents	Kódovník měřených veličin (např. pH, chloridy, úhrn měsíčních srážek)
Observations	Jednotlivá měření vázaná ke vzorkovanému hloubkovému intervalu (tj. např. hodnoty hladiny podzemní vody či pH)
Point Values	Jednotlivá měření vázaná ke konkrétní hloubce ve vrtu (tj. např. karotáž)

1.3. XML Formát

XML, neboli Extensible Markup Language, je rozšiřitelný značkovací jazyk. Je to univerzální formát, který je hojně používaný jako výměnný formát. Spousta aplikací jej využívá pro export dat, nebo jako formát pro uložení nastavení.

V XML se označuje význam jednotlivých části textu pomocí značek (tagů). Tagy se nachází mezi špičatými závorkami a jsou párové, tudíž každý počáteční tag musí být ukončen ukončovacím tagem. Informace jsou uloženy mezi počátečním a koncovým tagem, nebo pomocí atributů, které jsou definovány u počátečního tagu ve špičaté závorce za názvem tagu. Příklad párového tagu s atributem: `<nazev_tagu atribut='hodnota_atributu'> hodnota tagu </nazev_tagu>`.

Zpracování XML podporuje řada nástrojů a programovacích jazyků, ve kterých často existují knihovny pro práci s daty v tomto formátu.

1.4. Action sequences

Action sequences jsou XML dokumenty, které obsahují definice základních úloh. Zajišťují spuštění obsahu vytvořeného v platformě Pentaho Business Inteligence. Jsou kódované ve specifickém XML formátu a obvykle jsou uloženy v souborech s příponou „.xaction“. [3]

Action sequences mohou být použity například pro zavolání Pentaho reportu, spuštění transformace, nebo k odeslání emailu. Pro jejich návrh se používá nástroj Pentaho Design Studio.

V této práci jich mohlo být využito pro generování reportů vrtných profilů a následné rozesílání těchto reportů uživatelům. V tomto případě však bylo možné využít komponent nástroje Pentaho Data Integration. Generování a rozesílání reportů je tak součástí datových transformací a řešení úkolu je tak celistvější.

2. Použité prostředky

2.1. Pentaho Data Integration (Kettle)

Pentaho Data Integration (PDI), neboli Kettle, je nástroj od Pentaha, zajišťující ETL – procesy čtení, transformace a nahrávání dat. Díky ETL nástrojům je nejčastěji používaný v prostředí datových skladů, ale může být použit i pro jiné účely (např. migrace dat mezi aplikacemi, export dat z databází, import dat do databází, atd.). [1]

PDI může být použito jako samostatná aplikace, nebo jako část větší sestavy Pentaho Suite. Je to nejpopulárnější dostupný open source ETL nástroj. PDI podporuje širokou škálu vstupních i výstupních formátů včetně textových souborů a komerčních i volně přístupných databázových modulů. Kromě toho transformační možnosti PDI umožňují manipulovat s daty jen s velmi málo omezeními. [4]

Každý proces je vytvořen pomocí grafického prostředí, kde uživatel specifikuje, co má daný proces dělat, aniž by musel psát kód.

V PDI se dají vytvářet dva typy procesů:

- Transformace - Používají se pro převod vstupních dat na požadovaný výstup.
- Job - Slouží k propojení dílčích transformací, obsahují jiné funkce než transformace (např.: odesílání emailu při chybě v transformaci)

Transformace se vytváří jako diagram z předdefinovaných kroků (anglicky step), které se mezi sebou propojují. Každý krok má své nastavení, pomocí kterého se definuje jeho funkcionalita.

2.2. Pentaho Report Designer

Pentaho Report Designer je grafický nástroj, pro vytváření a úpravu reportů. V tomto nástroji se definuje grafická šablona pro zobrazení dat a jejich zdroj. Zdrojem dat může být například databáze (MS SQL, MySQL, PostgreSQL, Oracle, atd.), transformace vytvořená v Pentaho Data Integration, tabulka, nebo XML soubor. Šablona pro zobrazení se vytváří pomocí grafického prostředí, ve kterém uživatel definuje vzhled a rozložení reportu pomocí komponent (label, text-field, image, atd.).

2.3. PostgreSQL

PostgreSQL (nebo zkráceně Postgres) je plnohodnotným relačním databázovým systémem s otevřeným zdrojovým kódem. Má za sebou více než devatenáct let aktivního vývoje a má vynikající pověst pro svou spolehlivost i bezpečnost. Běží nativně na všech rozšířených operačních systémech včetně Linuxu, UNIXů a Windows. PostgreSQL má rozhraní pro mnoho programovacích jazyků (např. C/C++, Java, .Net, Perl, Python, Ruby a mnoho dalších). [5]

2.4. Python

Python je interpretovaný, objektově orientovaný, vysokoúrovňový programovací jazyk s dynamickou sémantikou. Obsahuje vysokoúrovňové datové struktury a zároveň se jedná o dynamický programovací jazyk[1], což z Pythonu dělá velmi oblíbený programovací jazyk pro rychlý vývoj aplikací (Rapid Application Development). [6]

Interpret[2] pro Python, stejně jako mnoho rozšiřujících knihoven, je dostupný pro všechny nejrozšířenější platformy (Linux, Windows, MacOS). [6]

[1] Dynamický programovací jazyk vykonává za běhu operace, které statické programovací jazyky provádí již při kompilaci.

[2] Počítačový program, který umožňuje spouštět jiný program přímo ze zdrojového kódu, napsaného ve zvoleném programovacím jazyce, a není ho nutné převádět do strojového kódu.

3. Instalace a nastavení prostředků

3.1. Příprava PostgreSQL

3.1.1. Instalace PostgreSQL

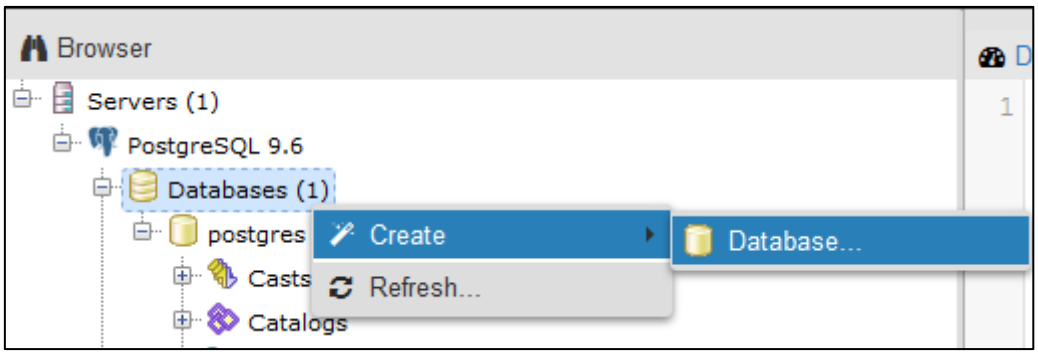
Pro stažení instalačního souboru jsem použil odkaz z oficiálních stránek PostgreSQL - https://www.postgresql.org/download/windows/. Dále stačilo vybrat verzi PostgreSQL (zvolil jsem nejnovější verzi - PostgreSQL 9.6.2) a operační systém (Windows x86-64).

Samotná instalace probíhá v průvodci. Volí se při ní mimo jiné heslo pro administrátorský účet (superuser) a také port, na kterém bude server naslouchat. Dále již proběhne samotná instalace.

Po skončení instalace PostgreSQL serveru, je možné pomocí průvodce Stack Builder doinstalovat rozšíření a podpůrné programy. Pro potřeby této práce to však není nutné. Součástí základní instalace je grafické prostředí zvané *pgAdmin* a také textové prostředí *SQL Shell (psql)*. V obou prostředích je možné spravovat databázi.

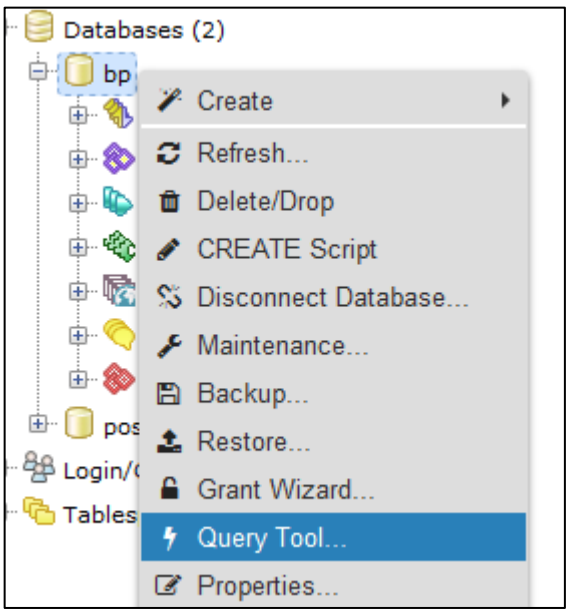
3.1.2. Vytvoření databáze

Pro vytvoření nové databáze a import struktury databáze jsem využil grafické prostředí *pgAdmin 4*. Toto prostředí obsahuje navigační panel, ve kterém najedu na Položku *Databases* a pravým kliknutím na ní vytvořím novou databázi (viz obrázek 2) s názvem „bp".



Obrázek 2 - Vytvoření nové databáze

Po vytvoření databáze je třeba vytvořit její strukturu (tabulky, klíče, relace, popisy, atd.). Pro vytvoření struktury databáze mi byl poskytnut vedoucím práce vyexportovaný SQL příkaz v textové podobě. Spuštění tohoto SQL kódu zajistilo vytvoření všech tabulek včetně nastavení klíčů, relací a popisů. Pro zadání SQL příkazu, určeného pro určitou databázi, je třeba v navigačním panelu kliknout pravým tlačítkem na tuto databázi a zvolit nástroj *Query Tool (*viz obrázek 3*)*. Do textového pole v tomto nástroji se následně vloží SQL příkaz a potvrdí se tlačítkem *Execute*.



Obrázek 3 - Nástroj Query Tool

3.2. Instalace Javy

Nástroje Pentaho jsou napsány v „Javě" a tak stejně jako mnoho dalších aplikací potřebují pro své spuštění a chod běhové prostředí Javy (Java Runtime Environment). To je možné stáhnout z oficiálních stránek Javy na adrese https://java.com/en/download/.

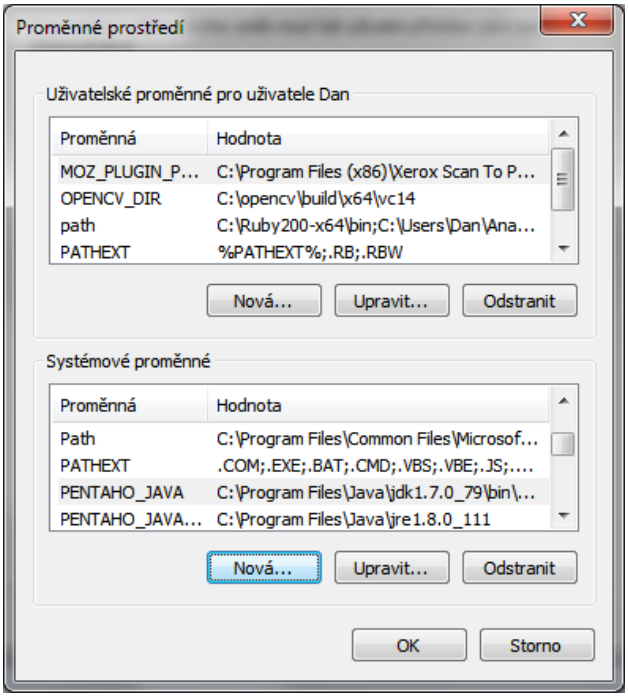
3.3. Příprava nástrojů Pentaho

3.3.1. Nastavení systémových proměnných

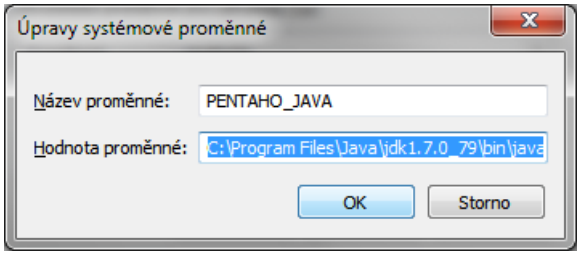
Oba použité nástroje (Pentaho Report Designer a Pentaho Data Integration) není třeba instalovat, pro běh těchto programů stačí mít nainstalované běhové

prostředí Javy (Java Runtime Environment) a nastavit dvě systémové proměnné „PENTAHO_JAVA_HOME“ a „PENTAHO_JAVA“.

Nastavení těchto proměnných provedeme v grafickém prostředí Windows 7 kliknutím na Start - Počítač - Vlastnosti - Upřesnit nastavení systému - Proměnné prostředí - Nová. Zde zadáme název proměnné a cestu k běhovému prostředí Javy, resp. cestu k souboru „java.exe“ (viz obrázek 4 a obrázek 5).



Obrázek 4 - Nastavení systémových proměnných



Obrázek 5 - Přidání systémové proměnné

3.3.2. Stažení a spuštění

Odkazy na stažení obou nástrojů jsou přímo na webových stránkách Pentaha. Pro PDI je to: http://community.pentaho.com/projects/data-integration/. Pro Report Designer: http://community.pentaho.com/projects/reporting/.

Po nastavení systémových proměnných stačí pouze rozbalit stažené soubory. Pro spuštění programů slouží dávkové soubory (.bat), které po rozbalení souborů najedeme přímo v adresářích nástrojů. Grafické prostředí PDI (Spoon) se spouští pomocí souboru „Spoon.bat“ a Report Designer pomocí „report-designer.bat“.

4. Vlastní řešení

4.1. Transformace v PDI

Hlavní část práce je tvořena transformacemi vytvořenými v grafickém prostředí PDI zvaném Spoon. Práce se skládá ze čtyř dílčích transformací, které jsou propojeny v jednom „Jobu" do funkčního celku, který dokáže kompletně zpracovat data, vytvořit z nich report a zaslat ho uživateli na email.

Transformace se skládají z předdefinovaných komponent (elementů). V následující tabulce (viz tabulka 3) je přehled komponent používaných v této práci a jejich popis.

Tabulka 3 - Komponenty PDI

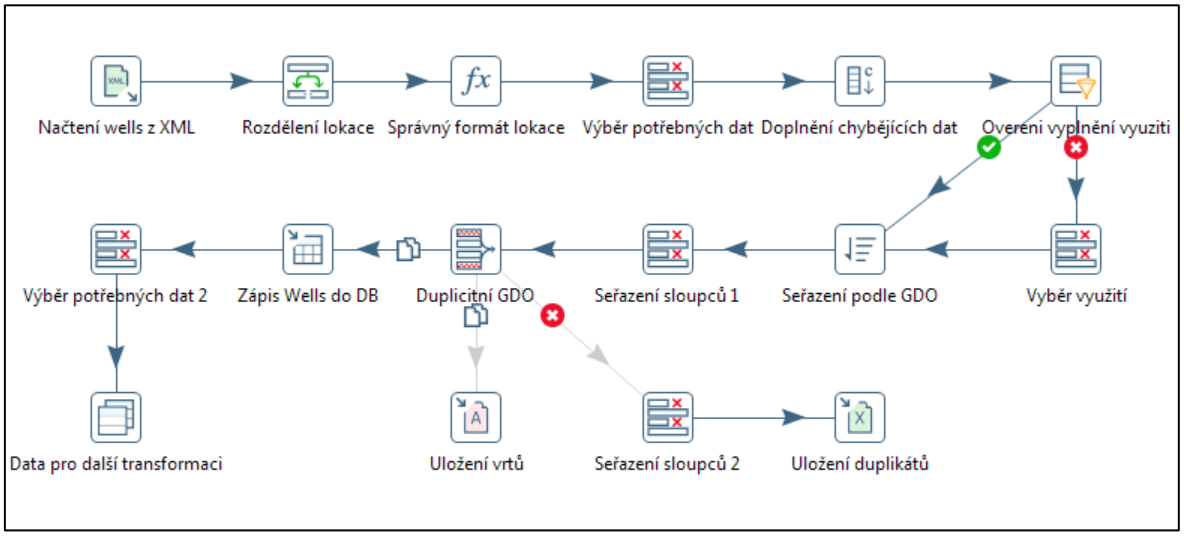
Náhled	Název komponenty	Popis komponenty
	Add Constant	Slouží k přidání konstant. Zadává se zejména datový typ konstanty a její hodnota.
	Analytic Query	Slouží k přístupu k datům v jiných řádcích. Můžeme tak data z několika řádků převést do jednoho řádku s několika novými sloupci. Data je možné seskupit podle vybraných sloupců, aby se slučovala pouze data, která mají něco společného.
	Concat Fields	Sloučí obsah několika sloupců do jednoho nového. Slouží tak například ke spojení textových řetězců, které jsou ve více sloupcích.
	Copy rows to result	Zajišťuje převod dat do další transformace v rámci „jobu".
	Filter rows	Tato komponenta obsahuje podmínku. Datový tok z této komponenty je rozdělen na dvě větve podle splnění/nesplnění podmínky.
	Formula	Tato komponenta se dá použít pro jednoduché výpočty (např.: sloupec_1 * sloupec_2) nebo pro jednoduché podmínky.

	Get data from XML	Načítá data z XML souborů. Je možné přímo zvolit datové typy vstupních dat a jejich formát (např. u desetinných čísel). Obsahuje funkci pro automatické načtení sloupců podle *tagů* v XML souboru. Pokud požadovaná vstupní data nejsou textem mezi počátečním a ukončovacím *tagem*, ale například atributem *tagu*, je nutné přidat je ručně.
	Get rows From result	Zajišťuje načtení dat z transformace, na kterou je tato napojena v rámci „jobu“. Aby bylo možné data načíst, musí být předchozí transformace obsahovat komponentu „Copy rows to result“. V této komponentě se pak nastavují názvy sloupců a jejich datové typy.
	Group by	Tato komponenta umožňuje počítat hodnoty pro skupiny dat. V nastavení se vyberou sloupce, které definují skupinu. Počítat se může například průměr, suma, minimum, maximum, atd.
	Mail	Umožňuje posílat email na předem definované adresy. Je nutné nastavit SMTP server, kterým může být například gmail a ověření. Text emailu musí být také předem definovaný. K emailu je možné přidat přílohu, jejíž adresa (v souborovém systému) se nastavuje přímo v této komponentě.
	Modified Java Script Value	Umožňuje vytvářet javascriptové výrazy. Ty mohou být použity například pro upravování textových řetězců (cesty k souborům, ..). Ale má i spoustu dalších použití.
	Pentaho Reporting Output	Tato komponenta umožňuje vytvářet reporty, podle předem definované šablony vytvořené v Pentaho Report Designer. Cesta k šabloně a k souboru, který má být z této šablony vytvořen, musí být dopředu definována. V nastavení komponenty se vybírá formát výstupního reportu (např.: PDF, HTML, atd.).

	Replace in string	Slouží k úpravám textových řetězců. Lze použít regulárních výrazů, nebo přímo hledací funkce, kterou tato komponenta obsahuje. Nahrazení hledané části je možné za jiný textový řetězec, nebo za hodnotu některého ze sloupců.
	Select values	Slouží k vybírání, odstraňování a přejmenovávání sloupců. Lze v ní také nastavovat délku textových řetězců, počty desetinných míst u reálných čísel, apod.
	Sort rows	Umožňuje seřadit data podle vybraného sloupce nebo několika sloupců. Podle nastavení může řadit vzestupně nebo sestupně.
	Split Fields	Rozdělí textový řetězec z nějakého sloupce do více sloupců podle zvoleného děliče. Tím může být například znak, nebo jiný textový řetězec.
	Table output	Tato komponenta slouží pro vkládání dat do SQL databáze. Obsahuje průvodce pro vytvoření připojení k databázi. Dále je třeba vybrat cílové schéma a tabulku v databázi. Dokáže také vrátit automaticky generované *id* záznamu.
	Unique rows	Odstraní duplicitní data, nastavuje se sloupec, nebo sloupce, při jejichž shodné hodnotě u více řádků se zachová pouze první záznam. Před tímto krokem musí být data seřazena pomocí komponenty „Sort rows“.

4.1.1. Transformace „transform_wells“

Tato transformace (viz obrázek 6) zajišťuje zpracování dat o hydrogeologických vrtech. Načítá data z XML souborů, převádí je do požadovaného formátu, ukládá do SQL databáze, a předává data další transformaci.



Obrázek 6 - Transformace „transform_wells“

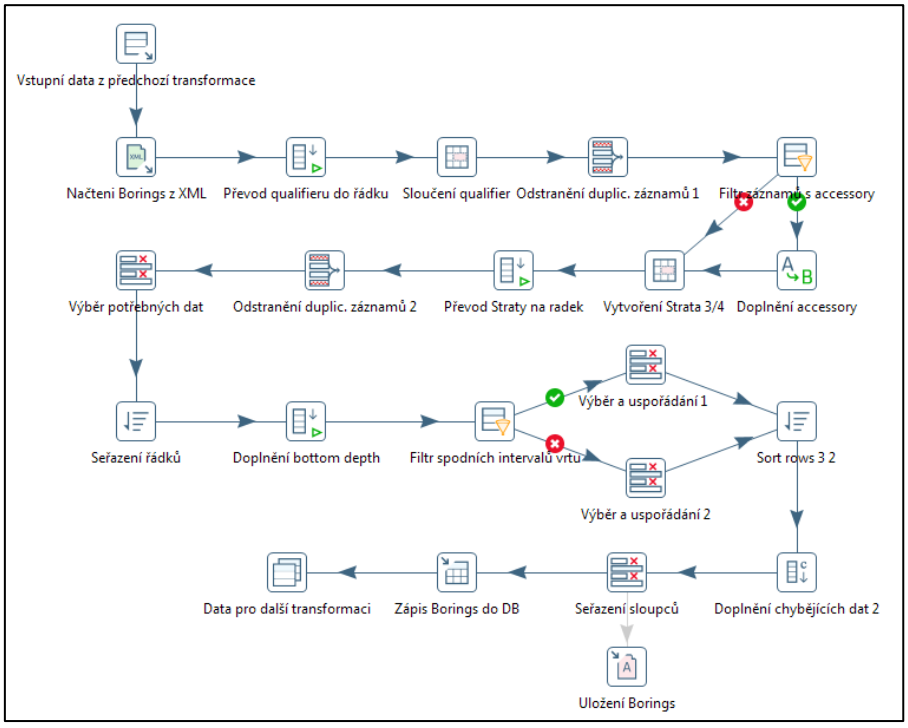
Popis důležitých komponent:

- **„Načtení wells z XML“** - Zajišťuje načtení dat týkajících se vrtů z XML souborů. Definuje se zde cesta ke vstupním souborům, dále pak datové typy a XPath načítaných dat. PDI dokáže načíst elementy, které XML soubor obsahuje, nedokáže však načíst atributy elementů, ty se musí doplnit ručně.
- **„Rozdělení lokace“** a **„Správný formát lokace“** - Rozdělení souřadnic do dvou polí a převod do správného formátu.
- **„Ověření vyplnění využití“** - Slouží jako podmínka, která filtruje záznamy, ve kterých nebyla vyplněna hodnota sloupce „vyuziti“. Rozděluje tok dat, podle splnění podmínky, pokud není splněna podmínka, putují data do komponenty „Výběr využití“, kde se místo vstupní hodnoty přepíše na výchozí hodnotu.
- **„Seřazení podle GDO“** a **„Duplicitní GDO“** - Seřadí řádky podle klíče GDO a následně odstraní duplicitní záznamy (záznamy, které mají stejný klíč GDO). Z této komponenty vede vypnutá větev, která obsahuje funkční bloky pro zápis do XLS souborů. Pokud se tato větev povolí, zapíšou se duplicitní záznamy do Excelové tabulky.

- **„Uložení vrtů“** - Vstup do této komponenty je vypnut. Slouží k uložení dat do databázového souboru aplikace MS Access. Může sloužit pro ukládání dat v případě nefunkčního připojení k SQL databázi.
- **„Zápis Wells do DB“** - Zapíše zpracovaná data o vrtech do SQL databáze, do tabulky „wells“. Je třeba definovat připojení k SQL databázi a tabulku, do které se mají záznamy zapisovat.

4.1.2. Transformace „transform_borings“

Tato transformace (viz obrázek 7) zpracovává záznamy o složení horniny v jednotlivých vrstvách vrtu. Z předchozí transformace přijímá názvy souborů, z nichž byly zpracovány údaje o vrtech, a dále z těchto souborů načítá informace o vrstvách horniny. Tyto informace stejně jako první transformace převede do požadovaného formátu a zapíše do databáze, následně opět předává data další transformaci.

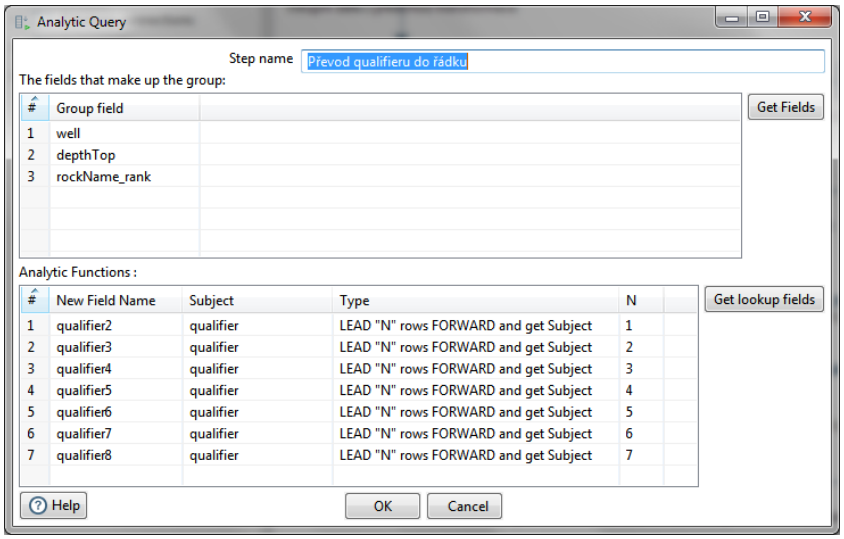


Obrázek 7 - Transformace „transform_borings“

Popis důležitých komponent:

- **„Načtení borings z XML“** - Zajišťuje načtení dat o složení horniny ze souborů (funguje podobně jako „Načtení wells z XML“ v předchozí transformaci).

- **„Převod qualifieru do řádku“** - Seskupí data podle vybraných sloupců, následně převádí popis horniny (qualifier) z předchozích řádků do nového pole v následujícím řádku (viz obrázek 8). Tím dojde k převedení celého popisu horniny, který je po načtení ze souborů v mnoha řádcích, do jednoho řádku a lze s ním dále pracovat.



Obrázek 8 - Převod qualifieru do řádku

Popis důležitých komponent:

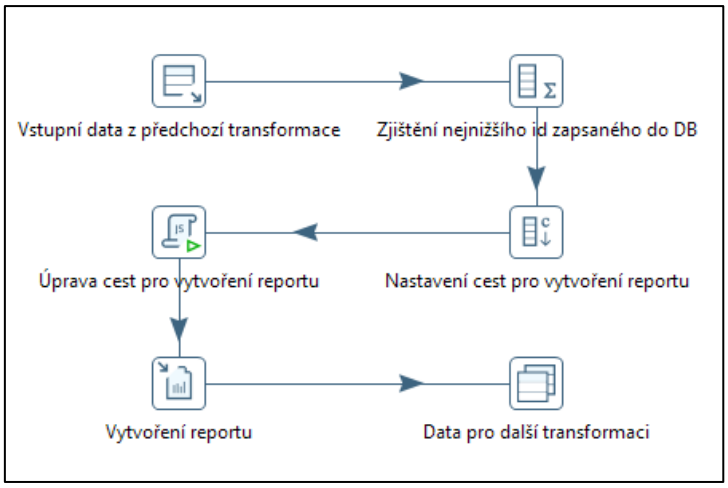
- **„Sloučení qualifier“** - Sloučí všechny záznamy o popisu horniny, převedené předchozím krokem do mnoha sloupců v jednom řádku.
- **„Odstranění duplic. záznamů“** - Odstranění neúplných duplicitních záznamů, vzniklých v předchozích komponentách, při převodu dat do řádku.
- **„Filtr záznamů z accessory“** - Pokud řádek obsahuje záznam o příměsích (accessory), bude následně doplněn textový řetězec „příměs: “ před hodnotu „accessory“.
- **„Vytvoření Strata 3/4“** - Sestavení celého popisu hlavní a vedlejší složky dané vrstvy vrtu. Popis se skládá z názvu horniny, doplňujícího popisu horniny (qualifier), barvy a příměsí.
- **„Doplnění bottom depth“** - Jelikož vstupní data obsahují pouze počáteční hloubku dané vrstvy vrtu, je nutné přidat také spodní hloubku. Pro tu je použita počáteční hloubka dalšího záznamu.
- **„Filtr spodních intervalů“** - Předchozí krok doplní spodní hloubku vrstvy, poslední záznam u každého vrtu však již nemůže brát hloubku z následujícího záznamu a nemá tak vyplněnou tuto hodnotu. Tato komponenta rozlišuje, záznamy, které nemají tuto hodnotu vyplněnou,

a pošle je do komponenty, která místo této hodnoty použije celkovou hloubku vrtu.

- **„Zápis Borings do DB“** - Zapíše zpracované informace o složení horniny v jednotlivých vrstvách do SQL databáze, do tabulky „borings“. Je třeba definovat připojení k SQL databázi a tabulku, do které se mají záznamy zapisovat.

4.1.3. Transformace „transform_reporting“

Úkolem této transformace (viz obrázek 9) je zajistit vytvoření reportu (viz 0) ze zpracovaných dat a jeho uložení v podobě PDF souboru do předem definovaného adresáře. Vstupem jsou data z předchozí transformace, ze kterých vybere nejnižší *id* vrtu a to použije jako parametr pro vytvoření reportu ze šablony. Nakonec opět posílá data další transformaci.

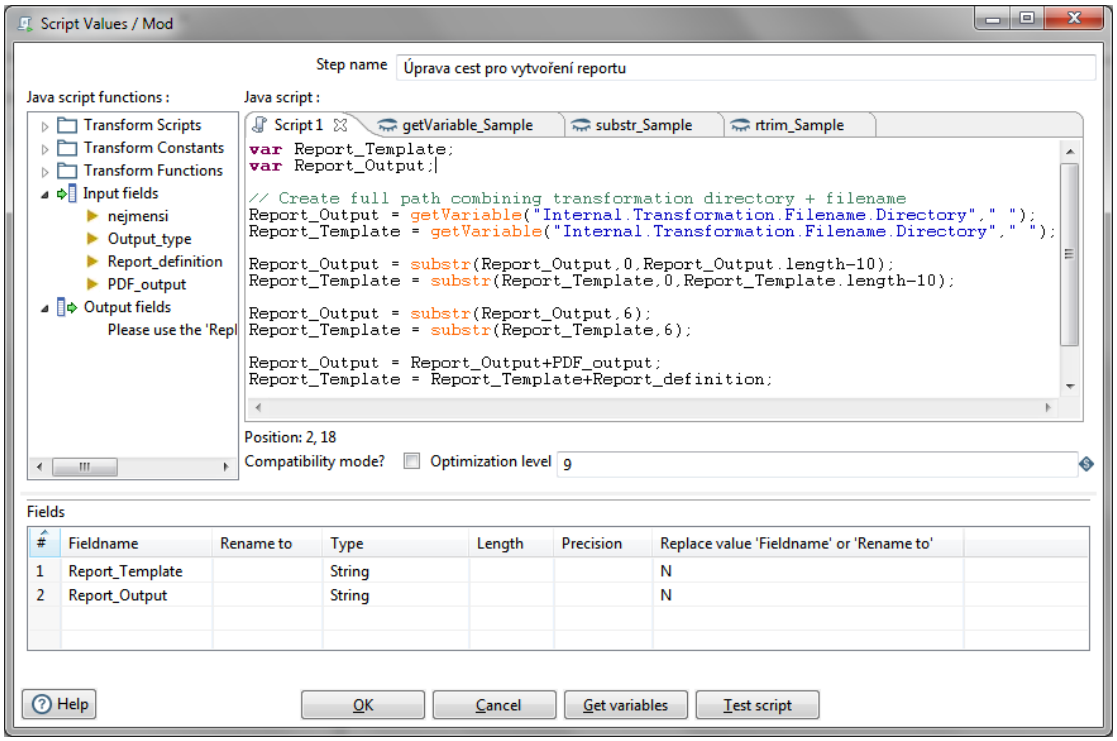


Obrázek 9 - Transformace „transform_reporting“

Popis důležitých komponent:

- **„Zjištění nejnižšího id zapsaného do DB“** - Tato komponenta zajišťuje zjištění nejnižšího zapsaného *id*. Nejnižší *id* je třeba pro generování reportu.
- **„Nastavení cest pro vytvoření reportu“** - Zde jsou zadány cesty a názvy souborů pro vytvoření reportu. Je zde definována relativní cesta k definičnímu souboru pro vytvoření reportu, relativní cesta k výstupnímu PDF souboru a výstupní formát reportu (PDF).
- **„Úprava cest pro vytvoření reportu“** - Tato komponenta pomocí javascriptu zajišťuje vytvoření cesty k souborům z relativních cest zadaných

předchozím krokem. Nejprve se z proměnné Pentaha zjistí cesta k transformaci. Dále je z ní pomocí řetězcových operací vytvořena cesta ke složce, která je v souborovém systému o úroveň výše. Nakonec jsou k adrese této složky přidány relativní cesty pro definiční soubor reportu a pro výstupní PDF soubor (viz obrázek 10). Vytváření cesty pomocí textových operací může vypadat poměrně složitě, ale bohužel je v Pentaho obtížné zajistit vytvoření cesty ke složce, která je uložena o úroveň výše, než samotná transformace. Pokud by byl report ve stejné složce, jako transformace, nebo v podsložce této složky, bylo by vytvoření cesty jednoduché, ovšem souborová struktura by byla nepřehledná.

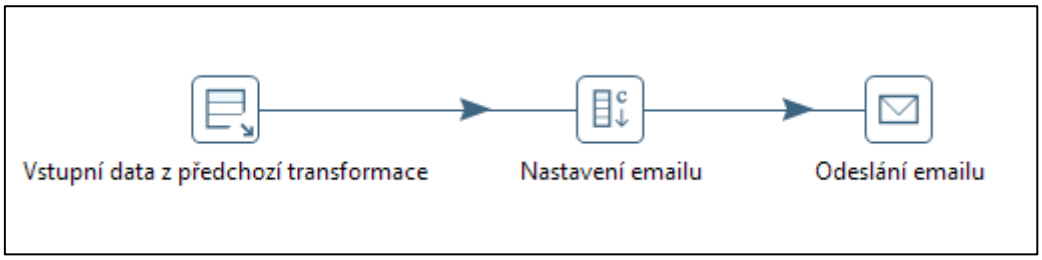


Obrázek 10 - Nastavení komponenty pro úpravu cest

- **„Vytvoření reportu“** - Tato komponenta zajišťuje vytvoření reportu z definičního souboru. Nastavuje se zde sloupec, ve kterém je cesta k definičnímu souboru, sloupec s cestou k výstupnímu PDF souboru, výstupní formát a atributy, které jsou pro vytvoření reportu třeba. Ke každému atributu je přiřazen sloupec, ve kterém jsou data pro tento atribut.

4.1.4. Transformace „transform_mail“

Tato transformace (viz obrázek 11) zajišťuje odeslání reportu vytvořeného předchozí transformací uživateli prostřednictvím emailu.



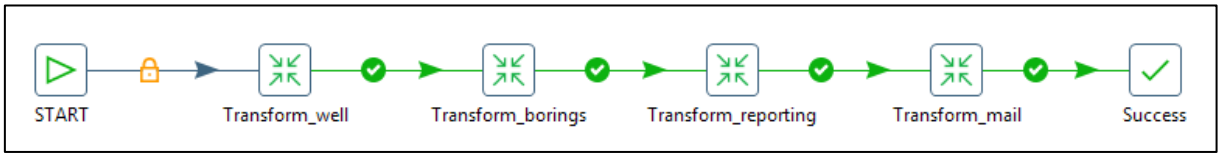
Obrázek 11 - Transformace „transform_mail“

Popis důležitých komponent:

- **„Nastavení emailu“** - V této komponentě se doplní informace potřebné pro připojení k SMTP serveru a také další data jako například adresa odesílatele, adresy příjemců, předmět zprávy, text zprávy, atp.
- **„Odeslání emailu“** - Tato komponenta zajišťuje připojení k SMTP serveru, kterým je v mém případě SMTP server od Googlu (gmail) a dále zajištuje samotné odeslání emailu, jehož přílohou je report vrtného profilu, uživateli.

4.1.5. Job „job_wells.kjb“

Tento *job* (viz obrázek 12) propojuje transformace, které řeší dílčí části této úlohy. Zajišťuje tak spuštění všech částí společně.



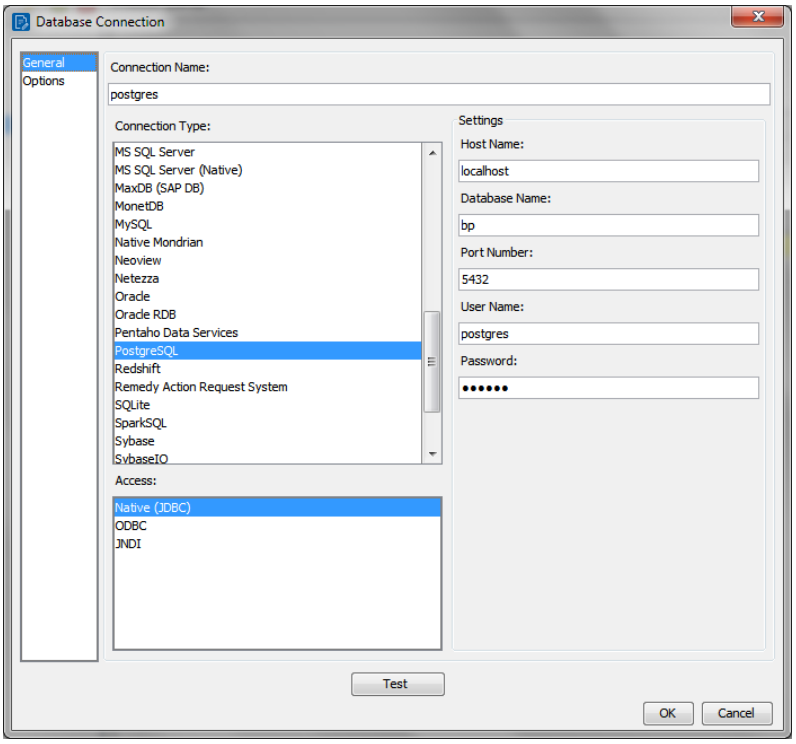
Obrázek 12 - Job „job_wells.kjb“

4.2. Návrh reportu v Pentaho Report Designer

4.2.1. Nastavení zdroje dat

Nejprve bylo nutné nastavit zdroj, ze kterého se mají brát data. Tímto zdrojem je SQL databáze PostgreSQL.

Přidání připravené databáze, jakožto nového zdroje dat, je možné kliknutím na Data - Add datasource - JDBC. Dále klinutím na zelený plus dojde k otevření formuláře pro vytvoření nového připojení k databázi. Zde je nutné vybrat typ databáze a nastavit přihlašovací údaje (viz obrázek 13).



Obrázek 13 - Připojení k Postgre SQL

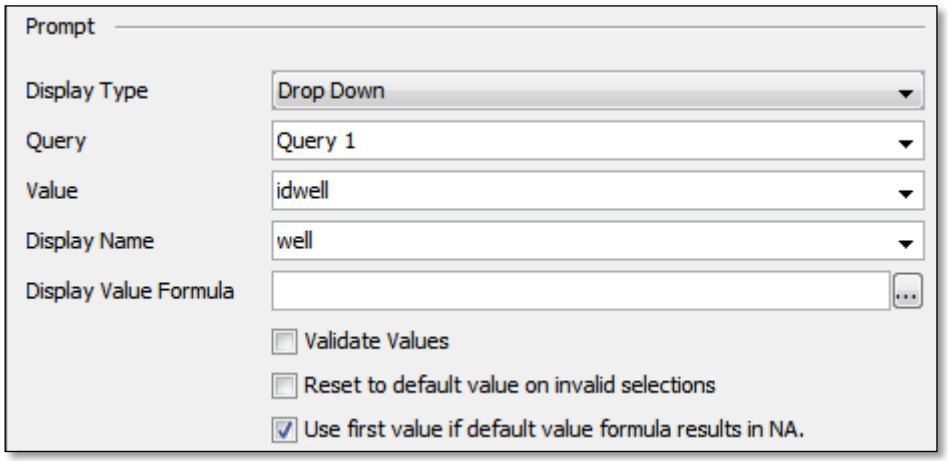
Dále je třeba zadat SQL dotazy, které nám z databáze vyberou potřebná data pro zobrazení v reportu a také data, z nichž bude možné zvolit vstupní parametr pro výběr vrtu, z jehož dat se report sestaví.

- 1. SQL dotaz bude sloužit pouze jako alternativní vstup atributu, kterým je *id* vrtu. Vybere z databáze pouze *id* a *klíče GDO* všech uložených vrtů. Je potřebný zejména pro následné nastavení vstupních parametrů a je využíván při vytváření reportu jiným způsobem, než spuštěním celé transformace, která je výsledkem této práce (například přímým spuštěním z definičního souboru).

- 2. SQL dotaz načítá z databáze veškerá data, která se v reportu zobrazují. Načte data o vrtech, jejichž *id* je větší nebo stejné jako vstupní parametr. To zajistí podmínka: „WHERE idwell >= ${id_well}“, kde *idwell* je identifikátorem vrtu v databázi a *id_well* je vstupní parametr.

Následně je ještě třeba nastavit samotný vstupní parametr. Pro vytvoření reportu pomocí transformace je potřeba definovat pouze název a datový typ parametru, tím ho bude možné načíst přímo v prostředí PDI. Nastavil jsem však i možnost vybrat vstupní parametr ze seznamu, pro možnost spuštění tvorby reportu přímo v prostředí Pentaho Report Designer.

Parametr se přidává kliknutím na Data - Add parametr. Zde zadáme název parametru („id_well“), dále vyplníme zobrazovanou hodnotu atributu („well“) a zvolíme datový typ („Integer“). Další nastavení (viz obrázek 14), je pouze pro možnost vytvoření reportu bez spuštění transformace. Nastavuje se zde forma výběru, vstupní dotaz, hodnota pro parametr a zobrazovaná hodnota.



Obrázek 14 - Nastavení parametru

4.2.2. Grafický návrh reportu

Samotný návrh reportu byl vytvořen pomocí grafického prostředí nástroje Pentaho Report Designer. Návrh se skládá z předdefinovaných komponent, kterými jsou v této práci zejména *label*, *text-field*, *message-field* a *band*.

- Label je použit jako textový popis (nadpis) pro proměnná data.
- Text-field slouží pro zobrazení proměnných dat.

- Message-field kombinuje použití *Label* a *Text-field*, slouží k zobrazení popisku, do kterého mohou být vložena proměnná data. Název proměnné, ze které chceme zobrazit data, musí být v závorce, před kterou je znak dolaru. Příklad zobrazení proměnné v *message-field*: `$(název_proměnné)`
- Band seskupuje několik komponent dohromady a slouží k snazšímu formátování.

Pro přehlednější zobrazení více řádků tabulky je dobré použít pro liché a sudé řádky různou barvu. V této práci se to týkalo zejména záznamů o složení horniny v jednotlivých vrstvách, kde každá vrstva byla na jednom řádku. Pentaho Report Designer obsahuje funkci pro automatické nastavení různé barvy pozadí lichých a sudých řádků pro komponentu *band*.

Stačilo tak dát všechny komponenty zobrazující data o vrstvách do jednoho „bandu" a nastavit mu atribut *name* na nějakou unikátní hodnotu. Dále pak bylo potřeba tuto funkci přidat. Přidání této funkce se provede kliknutím na: *Data - Add Functions - Report - Row Banding*. Přidané funkci pak již stačí nastavit, na jakou komponentu se má aplikovat, barvy pozadí a po kolika řádcích se má měnit barva pozadí. Na obrázku níže (viz obrázek 15) je zobrazeno nastavení této funkce, kde „borings_row" je název (hodnota atributu *name)* u komponenty *band*, která seskupuje data o vrstvě vrtu.

Name	Value
Required	
Function Name	RowBandingFunction0
Apply Element(s) Named	borings_row
Optional	
Number Of Rows	1
Active Banding Color	
Inactive Banding Color	

Obrázek 15 - Nastavení funkce

Ukázka výsledného reportu vrtného profilu s konkrétními daty je na následujícím obrázku (viz obrázek 16). Report má podobnou strukturu jako reporty, které jsou poskytovány z archivu Geofond Českou Geologickou Službou.

VRT 278470 - Základní informace o vrtu

ID Vrtu	278470	**Poskytovatel dat**	Česká geologická služba - Geofond
Původní název	S-2	**Provádějící organizace**	Stavoprojekt Hradec Králové
Hloubka Vrtu	5,90	**Druh objektu**	vrt svislý
Souřadnice X - JTSK [m]	1061050,0	**Účel vrtu**	inženýrsko-geologický
Souřadnice Y - JTSK [m]	621040,0	**Využití vrtu**	
Nadmořská výška	266,60	**Dokumentace**	GF V050345
Způsob zaměření X,Y	odečteno z mapy		
Hloubka hladiny podzemní vody	0,50		

Hloubka [m]	Stratigrafie	Popis hlavní složky Popis vedlejší složky
0.0 - 0.2	**Kvartér**	**písek jemnozrnný hlinitý humózní černá hnědá**
0.2 - 1.2	**Kvartér**	**písek jemnozrnný střednozrnný tmavá hnědá** štěrk ojediněle
1.2 - 2.5	**Kvartér**	**písek jemnozrnný šedá** štěrk zastoupení horniny - 30 % max.velikost částic 7 cm
2.5 - 2.8	**Kvartér**	**štěrk zastoupení horniny - 40 % max.velikost částic 1 dm**
2.8 - 3.2	**Kvartér**	**jíl písčitý šedá hnědá** štěrk ojediněle
3.2 - 3.9	**Kvartér**	**písek střednozrnný hnědá** štěrk zastoupení horniny - 25 % max.velikost částic 6 cm

Obrázek 16 - Report vrtného profilu

4.3. Transformace v Pythonu

Transformace implementovaná v Pythonu zpracovává data obdobně jako první dvě transformace vytvořené v PDI. Rozdíl mezi těmito transformacemi je ve způsobu uložení dat. Program v Pythonu neukládá data přímo do SQL databáze, ale do csv souboru, ze kterého je možno data snadno importovat do databáze.

Kód v Pythonu je členěn do funkcí. Načtená data si nejprve převádím na vlastní strukturu, se kterou se mi dále lépe pracuje, až následně samotná data zpracovávám. Kromě samotného zpracování dat obsahuje můj skript také funkce pro přehledný výpis zpracovávaných dat do konzole.

Zpracování dat v Pythonu se skládá z celkem patnácti funkcí a hlavního programu, který těchto funkcí využívá. Tři funkce zajišťují zápis dat do CSV souborů, dvě funkce jsou pro výpis a zbylých 8 funkcí zajišťuje samotné načtení a zpracování dat.

4.3.1. Funkce v Pythonu

- **Funkce „writeWellsToCSV“** zajišťuje zápis dat o vrtu do CSV souboru. Pokud již existuje soubor s daty, připíše pouze nová data na konec souboru. Když soubor ještě neexistuje, vytvoří nový, na první řádek vloží hlavičku a až následně vkládá data.
- **Funkce „writeBoringsToCSV“** zajišťuje zápis dat o složení horniny v jednotlivých vrstvách vrtu. Funguje podobně jako předchozí funkce („writeWellsToCSV“).
- **Funkce „fileExistsAndNotEmpty“** zjišťuje, zda vstupní soubor existuje a obsahuje již nějaká data. Tato funkce je používána oběma předchozími funkcemi pro zápis do souborů.
- **Funkce „getFileNames“**, jejímž vstupem je cesta ke složce se vstupními soubory, vrací názvy všech XML souborů, které se v dané složce nachází.
- **Funkce „getRootsOfXMLFile“** vrací kořenový element stromové struktury vstupního XML souboru.
- **Funkce „getDictionaryOfChilds“ a „getArrayOfIntervalDictionaries“** převádí načtená data do mé vlastní struktury, se kterou následně pracuji. Jsou to rekurzivní funkce, které prochází celou stromovou XML strukturu načtených souborů od kořene (root) a převádí ho na slovník (dictionary). V tomto slovníku je každý element uložen jako další slovník (resp. podslovník), který je v nadřazeném slovníku uložen pod klíčem, jímž je

název tohoto elementu. Text mezi počátečním a koncovým tagem elementu je ve slovníku vytvořeném z tohoto elementu pod názvem (klíčem) „elementText“, atributy jsou pak pod klíčem „elementAttributes“, což je opět slovník a každý atribut je v tomto slovníku uložen pod svým názvem. Příklad převedení XML stromu do mé struktury je na obrázcích níže (viz obrázek 17 a obrázek 18). Funkce „getDictionaryOfChilds“ vytváří novou strukturu pouze tam, kde se neopakují stejné elementy. Pro správné uložení hloubkových intervalů a informací o nich, tedy tam, kde se stejné elementy opakují, je tu druhá funkce („getArrayOfIntervalDictionaries“), která z intervalů vytvoří pole slovníků, jejichž obsah je opět zpracováván první funkcí.

```
<prvni value='attribut elementu'>
    <druhy value='atribut podelementu'>
        Text podelementu.
    </druhy>
</prvni>
```

Obrázek 17 - Příklad vstupní XML struktury

```
#Přístup k hodnotě atributu "value" elementu "prvni".
slovnik['prvni']['elementAttributes']['value']

#Přístup k hodnotě elementu "druhy".
slovnik['prvni']['druhy']['elementText']

#Přístup k hodnote atributu "value" elementu "druhy".
slovnik['prvni']['druhy']['elementAttributes']['value']
```

Obrázek 18 - Příklad přístupu k mé struktuře

- **Funkce „createWellDataDictionary“** vytváří slovník, který obsahuje všechna data o vrtu. Klíče slovníku jsou stejné jako názvy sloupců v databázi, je tak následně snadné uložit z tohoto slovníku data do CSV souborů, ze kterých je bude možné importovat do databáze. V této funkci se definuje, která vstupní hodnota patří ke kterému sloupci v databázi (klíči ve slovníku).
- **Funkce „createArrayOfBoringDataDictionaries“** vytváří pole slovníků, kde každý slovník obsahuje všechna data o vrstvě zkoumané horniny. Má podobný význam jako předchozí funkce. Obě tyto funkce při přiřazování hodnot používají čtyři následující pomocné funkce.
- **Funkce „splitAndEditLocation“** zajistí rozdělení lokace na „northing“ a „easting“ a převod těchto hodnot do správného formátu.

- **Funkce „setDefaultIfNull“** zjistí, zda vstupní hodnota existuje, a není to pouze prázdný textový řetězec. Pokud je nastavena, vrátí tato funkce vstupní hodnotu, jinak vrátí zadanou výchozí hodnotu.
- **Funkce „createRockDescription“** vytvoří celý popis vstupní vrstvy horniny. Tato poměrně jednoduchá funkce plní úkol mnoha bloků transformace vytvořené v PDI.
- **Funkce „getBottomDepthValue“** zajišťuje zjištění spodní hloubky vrstvy vrtu. Zjišťuje, zda je vstupní vrstva posledním záznamem u vrtu. Podle toho nastaví jako tuto hodnotu buď počáteční hloubku další vrstvy, nebo celkovou hloubku vrtu.
- **Funkce „printWells“ a „printBoringIntervals“** slouží k výpisu zpracovaných dat ze slovníků do konzole.
- **Hlavní program** používá předdefinované funkce. Na začátku programu jsou definovány konstanty, kterými jsou například cesty ke vstupním a výstupním souborům („INPUT_PATH“, „OUTPUT_PATH“) nebo pole obsahující pořadí názvů sloupců v databázi („SEQUENCE_OF_WELL_KEYS“, „SEQUENCE_OF_BORINGS_KEYS“). Dále program načte názvy všech souborů ve složce pro vstupní data, definované konstantou. Následně probíhá cyklus, ve kterém je každý soubor načten, převeden do mé struktury a pomocí ní je následně zpracováván. Po té, co jsou data ze souboru zpracována, vypíšou se do konzole a nakonec se zapíšou do výstupních CSV souborů. Na obrázku níže je ukázka kódu hlavního programu (viz obrázek 19).

```
266 fileNames = getFileNames(INPUT_PATH)
267 for wellFile in fileNames:
268     root = getRootOfXMLFile(INPUT_PATH + wellFile)
269     well = dict()
270     borehole = root[0]
271     dictOfChilds = getDictionaryOfChilds(root)
272     wellDataDict = createWellDataDictionary(dictOfChilds)
273     arrayOfBoringDataDictionaries = createArrayOfBoringDataDictionaries(dictOfChilds)
274 
275     print
276     print "-------------------------------------------------------------------"
277     print "---                  Informace o VRTU                           ---"
278     print "-------------------------------------------------------------------"
279     printWell(SEQUENCE_OF_WELL_KEYS, wellDataDict, 1)
280     print
281     print "-------------------------------------------------------------------"
282     print "---               Záznamy o složení horniny                     ---"
283     print "-------------------------------------------------------------------"
284     printBoringIntervals(arrayOfBoringDataDictionaries)
285 
286     csvFileOutputWell = OUTPUT_PATH + 'wells.csv'
287     csvFileOutputBorings = OUTPUT_PATH + 'borings.csv'
288 
289     writeWellsToCSV(wellDataDict, SEQUENCE_OF_WELL_KEYS, csvFileOutputWell)
290     writeBoringsToCSV(arrayOfBoringDataDictionaries, SEQUENCE_OF_BORINGS_KEYS, csvFileOutputBorings)
291 
```

Obrázek 19 - Část kódu hlavního programu

5. Porovnání PDI s Pythonem

V Pythonu jsem implementoval zpracování dat z XML souborů, jejich převod do požadovaného formátu a zápis do CSV souborů. Tato datová transformace tak řeší stejný úkol jako první dvě transformace vytvořené v PDI.

Tvorba transformací v Pythonu pro mě byla snazší a zejména časově rychlejší. Svůj vliv na to má i fakt, že s programováním v Pythonu mám více zkušeností, než s vývojem transformací v PDI. Není tak možné naprosto objektivně porovnat náročnost implementace.

Vzhledem k tomu, že v Pentahu řeším také další úkoly (vytvoření reportu vrtného profilu a rozesílání emailů uživatelům), je velkou výhodou mít i datové transformace implementované ve stejném nástroji, jako zbylé části práce.

5.1. Výhody PDI

- PDI obsahuje mnoho předdefinovaných funkčních bloků. Každý funkční blok po nastavení může řešit určitou úlohu. Některé funkční bloky, případně spojení několika funkčních bloků, mohou po poměrně rychlém a snadném nastavení řešit i poměrně komplikované a složité úkoly, jejichž řešení pomocí programovacího jazyka (např.: Pythonu) by bylo mnohem složitější a časově náročnější.
- PDI dokáže přehledně zobrazovat tok dat mezi jednotlivými komponentami. To je velká výhoda při vývoji transformací a zejména je toto výhodou při hledaní chyb.
- Velkou výhodou je také kompatibilita a snadné propojení datových transformací vytvořených v PDI s dalšími nástroji od Pentaha. Dobrým příkladem je snadné generování reportů vytvořených v Pentaho Report Designer pomocí komponenty *Pentaho Reporting Output*, která je obsažena v PDI.
- Je snadné mezi sebou propojovat dílčí transformace. To je velká výhoda při tvorbě větších projektů, které se skládají z mnoha dílčích transformací. Díky snadnému propojení je možné jednotlivé transformace vyvíjet paralelně.

5.2. Nevýhody PDI

- I přes velké množství předdefinovaných komponent není možné, aby existovala komponenta pro řešení jakékoliv problematiky. To občas znamená, že poměrně jednoduchý úkol musí být řešen dost složitě a výsledkem tak není vždy úplně elegantní řešení. Navíc při tomto problému může dojít k omezením (např.: pevně nastavené hodnoty), která tak pro některé případy mohou způsobit chybné zpracování dat.
- Nevýhodou je také poměrně složité detekování a řešení chyb ve vstupních datech. Pokud vstupní data budou mít odlišnou strukturu, než předpokládaná vstupní data, nastane chyba. V PDI je možné nastavit u každé komponenty, aby data, u kterých nastala chyba, pokračovala do jiné následující komponenty, než validní data. Ovšem i tak jsou zde omezené možnosti, jak tuto chybu řešit. Nejčastěji tak je možné pouze vytvořit a uložit záznam o chybě. V programovacím jazyce je možno vyvinout jednoduché uživatelské rozhraní, pomocí kterého bude moci uživatel určit, co se má dít při chybovém stavu (např.: pokračování dalším řádkem, ruční přepsání hodnoty, atd.).

6. Zhodnocení a závěr práce

Během vytváření transformací v PDI jsem narazil na dva problémy, které komplikovaly realizaci tohoto projektu. Jedním z nich je ověření validity XML souboru. Pokud budu zpracovávat více XML souboru najednou a narazím na nějaký, který není validní (např. neobsahuje ukončovací tag), v PDI nemám jinou možnost, než ukončit celou transformaci, případně nechat vypsat chybovou hlášku, ve kterém souboru nastala chyba. Druhým problémem jsou omezené prostředky pro vzájemnou práci s daty, která nejsou ve stejném řádku. V projektu jsem na tuto komplikaci narazil, když jsem potřeboval sloučit „qualifiery" (přídavná jména pro popis horniny), které jsou u stejného vrtu, ve stejné hloubce a ve stejné složce. Jediným prostředkem, který toto dokáže, je „Analytic Query", který ovšem vyžaduje přesné zadání odkud kam chci data přesouvat (ze kterého řádku, na který a do jakého sloupce). Zde jsem tak musel pevně nastavit, že pro popis horniny může být použito maximálně osm hodnot „qualifieru" (v testovaných datech bylo nejvýše 6, nejčastěji 2-3). Řešením tohoto problému a hledáním alternativ, jak by jej bylo možné řešit jinak, jsem se zabýval poměrně dlouho, ale nepodařilo se mi najít žádnou lepší variantu.

ETL transformace, vytvořené v rámci této práce, umožňují převést zdrojová hydrogeologická data z původního XML formátu do požadovaného výměnného formátu a zapsat je do databáze PostgreSQL. Další transformace vygeneruje report vrtného profilu z definičního souboru, který jsem vytvořil v Pentaho Report Designer. Poslední transformace pak odešle vygenerovaný report na email uživateli. Tyto transformace jsou propojeny v jednom „Jobu" a spouští se tak společně.

Dalším mým úkolem bylo porovnat tvorbu transformací v Pentaho Data Integration s vlastním řešením pomocí Pythonu. Vytvořil jsem skript v Pythonu, který je členěn do funkcí a dokáže zpracovat data o hydrogeologických vrtech ze vstupních souborů a zapsat je v požadovaném formátu do CSV souborů. Při porovnání implementace transformací v PDI a Pythonu nebylo možné jednoznačně určit, která z možností je lepší. Obě řešení mají své výhody i nevýhody a vždy je tak nutné zvážit, který nástroj je pro danou úlohu a za daných podmínek výhodnější.

Seznam použité literatury

1. Pentaho Data Integration (Kettle) Tutorial. *Pentaho Wiki* [online]. Orlando: Pentaho A Hitachi Group Company, 2008 [cit. 2017-05-14]. Dostupné z: http://wiki.pentaho.com/display/EAI/Pentaho+Data+Integration+%28Kettle%29+Tutorial
2. NEŠETŘIL, Kamil. Dokumentace HgIS. In: *Hydrogeologický informační systém: Správa a analýza dat o životním prostředí* [online]. Liberec: Technická univerzita v Liberci, 2016 [cit. 2017-05-14]. Dostupné z: http://www.hgis.cz/dokuwiki/doku.php?id=cs:documentation.
3. Action Sequences. *Pentaho Wiki* [online]. Orlando: Pentaho A Hitachi Group Company, 2008 [cit. 2017-05-11]. Dostupné z: http://wiki.pentaho.com/display/ServerDoc1x/03.+Action+Sequences.
4. CASTERS, Matt, Roland BOUMAN a JOS VAN DONGEN. *Pentaho Kettle solutions: building open source ETL solutions with Pentaho Data Integration*. [Online-Ausg.]. Indianapolis, IN: Wiley, 2010. ISBN 9780470635179.
5. PostgreSQL. *PostgreSQL* [online]. The PostgreSQL Global Development Group, ©1996-2017 [cit. 2017-04-24]. Dostupné z: https://www.postgresql.org/about/
6. What is Python? Executive Summary. *Python.org* [online]. Python Software Foundation, ©2001-2017 [cit. 2017-04-24]. Dostupné z: https://www.python.org/doc/essays/blurb/

Příloha A - Obsah přiloženého CD

Struktura adresářů na CD a jejich popis:

/Dokumentace

Obsahuje soubor „dokumentace.pdf“, což je text této bakalářské práce ve formátu pdf.

/Pentaho

Obsahuje mé řešení zadané problematiky pomocí nástrojů Pentaho. Tato složka obsahuje následující podsložky:

- Input - Do této složky se vkládají vstupní XML soubory.
- Output - Do této složky se uloží vygenerovaný report.
- Report - V této složce se nachází definiční soubor pro report (report_well.prpt).
- Transform - Zde se nacházejí všechny 4 dílčí transformace a job.

Pro spuštění transformací je třeba překopírovat všechny tyto podsložky do jedné složky v počítači a změnit nastavení tří komponent v transformacích. Změna je nutná v komponentách pro zápis do databáze (název databáze a heslo). Toto nastavení je třeba provést v transformaci „transform_well_1.ktr“ u komponenty „Zápis Wells do DB“ a v transformaci „transform_borings_2.ktr“ u komponenty „Zápis Borings do DB“. Následně je ještě třeba změnit nastavení pro odesílán emailů. To se nachází v transformaci „transform_mail_4.ktr“ v komponentě „Nastavení emailu“ a je zde třeba změnit nastavení SMTP serveru, zdrojovou adresu a cílové adresy.

/Python

Obsahuje transformaci implementovanou v Pythonu (soubor „transformace_python.py“) a další dva adresáře (input a output). Před spuštěním transformace v Pythonu stačí překopírovat tuto strukturu do jedné složky v počítači. Do složky „input“ se vkládají vstupní XML soubory, ve složce „output“ budou uloženy výstupní CSV soubory „wells.csv“ a „borings.csv“.

/SQL

Obsahuje textový soubor (sql.txt) s SQL příkazy, které vytvoří strukturu databáze (tabulky, klíče, relace, popisy, atd.).