

Nástroje pro migraci provozních dat

Bc. Ing. Jiří Závodský

Diplomová práce
2014



Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Bc. Ing. Jiří Závodský**
Osobní číslo: **A11516**
Studijní program: **N3902 Inženýrská informatika**
Studijní obor: **Informační technologie**
Forma studia: **kombinovaná**

Téma práce: **Nástroje pro migraci provozních dat**

Zásady pro vypracování:

1. Prostudujte téma migrace dat v databázových systémech.
2. Zpracujte přehled dostupných softwarových nástrojů pro migraci dat.
3. Uvedte na příkladech použití dostupných ETL nástrojů.
4. Popište možnosti přístupu k datům v databázích PostgreSQL z C-Sharp.
5. Popište tok dat v inventarizaci lesů a možnost použití nástrojů pro migraci dat.
6. Vytvořte návrh databáze pro ukládání měřených dat.
7. Vytvořte aplikaci pro import měřených dat inventarizace lesů do databáze.
8. Vytvořte uživatelskou dokumentaci aplikace pro migraci dat.

Rozsah diplomové práce:

Rozsah příloh:

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

Seznam odborné literatury:

1. ARGAWAL, Vidya Vrat a James HUDDLESTON. Databáze v C# 2008: průvodce programátora. Vyd. 1. Brno: Computer Press, 2009, 424 s. ISBN 978-80-251-2309-6.
2. LABERGE, Robert. Datové sklady: agilní metody a business intelligence. 1. vyd Brno: Computer Press, 350 s ISBN 978-80-251-3729-1.
3. LACKO L'uboslav. Business Intelligence v SQL Serveru 2008: reportovací, analytické a další datové služby. Vyd. 1. Brno: Computer Press, 456 s. ISBN 978-80-251-2887-9.
4. MATTHEW, Neil a Richard STONES. Beginning databases with PostgreSQL: from novice to professional. 2nd ed. Berkeley, CA: Apress, 2005. ISBN 15-905-9478-9.
5. MOMJIAN, Bruce a Richard STONES. PostgreSQL: introduction and concepts. 2nd ed. Boston, MA: Addison-Wesley, 2001, xxiv, 637 p. ISBN 02-017-0331-9.
6. RAINARDI, Vincent. Building a data warehouse with examples in SQL Server. Berkley, CA: Apress; Distributed to the book trade worldwide by Springer-Verlag New York, 2008. ISBN 15-905-9931-4
7. RALPH KIMBALL, Joe Caserta. The data warehouse ETL toolkit practical techniques for extracting, clearing, conforming, and delivering data. Indianapolis, IN: Wiley. ISBN 07-645-7923-1

Vedoucí diplomové práce:

Ing. Petr Šilhavý, Ph.D.

Ústav počítačových a komunikačních systémů

Datum zadání diplomové práce:

21. února 2014

Termín odevzdání diplomové práce:

20. května 2014

Ve Zlíně dne 21. února 2014

prof. Ing. Vladimír Vašek, CSc.

děkan



doc. Mgr. Roman Jašek, Ph.D.

ředitel ústavu

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové/bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomová/bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou/bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou/bakalářskou práci nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- beru na vědomí, že pokud bylo k vypracování diplomové/bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové/bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové/bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně

.....
podpis diplomanta

ABSTRAKT

Diplomová práce se zabývá návrhem technologie importu terénních dat národní inventarizace lesů. V teoretické části je vymezen pojem datového skladu a jsou popsány ETL procesy, které slouží k periodické migraci dat. Práce popisuje čtyři vybrané ETL nástroje - softwarové aplikace Clover, Apatar, Scriptella a SQL Workbench/J. Použití těchto ETL nástrojů je demonstrováno na příkladech importů a exportů dat. Praktickým výstupem diplomové práce je softwarová aplikace umožňující načtení dat ze zdrojových projektů (soubory MS Access) a jejich uložení do zvolených datových tabulek systému PostgreSQL.

Klíčová slova:

Datový sklad, import a export dat, ETL proces, ETL nástroj, kvalita dat, databázový systém PostgreSQL, inventarizace lesů.

ABSTRACT

This thesis deals with the design of the technology which is used to import field data originated within the Czech national forest inventory. Concepts of data warehouses and ETL processes serving to periodic data migration are described in theoretical part of the thesis. For demonstration purposes, four ETL tools - software applications Clover, Apatar, Scriptella and SQL Workbench/J, have been applied to import and export example data. A practical outcome of the thesis is a software application that allows loading data from source measurement projects (MS Access files) and storing them into selected data tables in the target PostgreSQL database.

Keywords:

Data warehouse, data import and export, ETL process, ETL tool, data quality, PostgreSQL database system, forest inventory.

Děkuji panu Ing. Petrovi Šilhavému, Ph.D. za čas a trpělivost, praktické rady a odborné vedení této diplomové práce.

OBSAH

ÚVOD.....	9
I TEORETICKÁ ČÁST.....	10
1 TEORIE DATOVÝCH SKLADŮ	11
1.1 OLTP SYSTÉMY	11
1.2 OLAP SYSTÉMY	11
1.3 DATOVÝ SKLAD.....	12
1.4 DATOVÉ TRHY	13
1.5 IMPORT A EXPORT DAT	13
1.6 ETL PROCESY.....	14
1.7 FÁZE ETL PROCESU	15
1.7.1 Extrakce.....	16
1.7.2 Transformace.....	16
1.7.3 Nahrání dat	17
1.8 PROBLÉMY ETL PROCESŮ	17
1.9 KONTROLA KVALITY DAT.....	19
2 PŘEHLED ETL NÁSTOJŮ	21
2.1 CLOVER ETL.....	21
2.1.1 Komponenty a sestavení transformačního grafu.....	22
2.1.2 Definice metadat	24
2.1.3 Definice připojení.....	25
2.1.4 Příklad importu dat aplikací Clover ETL.....	26
2.1.5 Příklad exportu dat aplikací Clover ETL	28
2.2 APATAR.....	30
2.2.1 Datové zdroje	30
2.2.2 Operace	30
2.2.3 Příklad exportu dat v aplikaci Apatar.....	31
2.3 SCRIPTELLA.....	33
2.3.1 Definiční soubor aplikace Scriptella	33
2.3.2 Příklad importu dat pomocí aplikace Scriptella	34
2.4 SQL WORKBENCH/J.....	36
2.4.1 Správa JDBC ovladačů v aplikaci SQL Workbench/J.....	36
2.4.2 Definice připojovacích profilů aplikace SQL Workbench/J	37
2.4.3 Import dat aplikací SQL Workbench/J	38
2.4.4 Příklad exportu dat aplikací SQL Workbench/J.....	39
2.4.5 Migrace dat datové tabulky aplikací SQL Workbench/J	41
3 NÁSTROJE PRO IMPORT A EXPORT DAT DATABÁZOVÉHO SYSTÉMU POSTGRESQL.....	43
3.1 PŘÍKAZ COPY.....	43
3.2 ROZŠÍŘENÍ DBLINK.....	44
3.3 IMPORT A EXPORT BINÁRNÍCH DAT.....	46
4 PŘÍSTUPY K DATABÁZOVÝM SYSTÉMŮM.....	48

4.1	ODBC	48
4.2	NPGSQL	50
II	PRAKTICKÁ ČÁST	52
5	NÁVRH DATABÁZE	53
5.1	SEZNAM DATOVÝCH TABULEK	54
6	APLIKACE PRO MIGRACI DAT	57
6.1	GRAFICKÉ UŽIVATELSKÉ ROZHRANÍ APLIKACE	57
6.1.1	Hlavní formulář	57
6.1.2	Otevření lokálně uloženého souboru zdrojových dat.....	58
6.1.3	Připojení k cílové databázi	59
6.1.4	Otevření souboru zdrojových dat ze systému managementu ploch	60
6.1.5	Výběr ploch pro import do databáze	61
6.1.6	Import numerických a textových dat ploch do databáze.....	62
6.1.7	Smazání vybrané plochy v cílové databázi	64
6.1.8	Nastavení přenosu dat	64
6.1.9	Výběr cílových tabulek pro import dat	66
6.1.10	Definice zdroje dat pro cílovou tabulku.....	67
6.1.11	Mapování sloupců mezi zdrojovou a cílovou datovou tabulkou	68
6.1.12	Import a export binárních souborů fotografií půdních sond	69
6.1.13	Nápověda k aplikaci	70
6.1.14	Ukončení práce s aplikací	71
6.2	PROGRAMOVÁ DOKUMENTACE APLIKACE	71
6.2.1	Třída PlotList	71
6.2.2	Třída AcSourcePlotList.....	72
6.2.3	Třída PgSourcePlotList	73
6.2.4	Třída PgTargetPlotList.....	75
6.2.5	Třída PgPhotoList	75
6.2.6	Třída ETLDefinition	77
6.2.7	Třída DataMigration.....	79
6.2.8	Třída Access.....	80
6.2.9	Třída Npg	83
6.2.10	Třída ApplicationSettings	85
	ZÁVĚR	86
	SEZNAM POUŽITÉ LITERATURY.....	88
	SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK.....	90
	SEZNAM OBRÁZKŮ	91

ÚVOD

Postupem doby s rozvojem výpočetní techniky a informatiky, výpočetní technika stále více proniká do dalších oborů a ani lesnictví není výjimkou. V roce 2011 bylo zahájeno měření druhého cyklu národní inventarizace lesů. V rámci toho projektu jsou pracovníky terénních skupin Ústavu pro hospodářskou úpravu lesů sbírána rozsáhlá data popisující stav lesních porostů na území České Republiky. Popis není zaměřen pouze na sběr dendrometrických veličin, ale jsou zjišťována také data popisující vlastnosti stanoviště, půdní charakteristiky a popis fytoocenózy. Tento sběr dat a jejich následné zpracování by se samozřejmě neobešlo bez aplikace počítačů a informačních technologií.

Výsledkem měření na plochách inventarizační sítě je řada datových souborů, obsahujících zjišťované parametry na inventarizačních plochách. Zatímco postup sběru a rozsah šetření byl již velmi dobře zpracován a popsán v metodikách sběru dat, technologie pro skladování takto sbíraných dat v centrálním datovém úložišti je nyní řešena.

Úkolem předkládané diplomové práce je pokusit se navrhnout pracovní schéma v centrální databázi pro ukládání části dat terénního šetření, které bude sloužit k vývoji softwarové aplikace pro import měřených dat ze souborů terénních projektů. Cílem je připravení jednoduché softwarové aplikace, kterou by tato data mohla být do centrální databáze migrována. Druhým cílem je seznámení se s technologií datových skladů a poznání některých existujících softwarových řešení pro migrace provozních dat do datových skladů. Tyto existující nástroje by mohly být vhodnou alternativou k vyvíjenému řešení migrační aplikace.

Při návrhu technologie pro skladování dat a jejich zpracování jsou upřednostňovány open source nástroje. Při skladování dat bylo rozhodnuto o použití databázového serveru PostgreSQL. Databáze založené na PostgreSQL nabízejí řadu možných rozšíření. První předností je možnost použití řady funkcí v rozšíření PostGIS, které umožňují prostorovou analýzu geografických dat, kterými data inventarizace již ze svého principu jsou. Druhou nezanedbatelnou předností databází PostgreSQL je možnost jejich integrace s nástrojem R pomocí rozšíření jazyka plr. Toto rozšíření nabízí možnost vytvářet uložené procedury v jazyce R a využívat tak nezměrného počtu knihoven a funkcí pro matematicko-statistickou analýzu a zpracování měřených dat, které prostředím jazyka R nabízí.

I. TEORETICKÁ ČÁST

1 TEORIE DATOVÝCH SKLADŮ

První část práce se zaměřuje na teoretický úvod do problematiky datových skladů a problematiky plnění datových skladů daty. Popisuje rozdíl mezi technologií OLTP systémů, které jsou obvyklým zdrojem dat datových skladů a architekturou OLAP určenou pro analýzy dat datových skladů. Tato část práce je také především zaměřena na popis ELT procesu, jakož to subsystému datového skladu, který je odpovědný za pravidelné plnění datového skladu daty. Popsány jsou jednotlivé fáze ELT procesu a možné problémy vznikající při přenosu dat mezi zdrojovými systémy a datovým skladem.

1.1 OLTP systémy

Technologie uložení dat v databázích můžeme rozdělit do dvou skupin a to na systémy transakční a analytické. Zkratkou OLTP (On-Line Transaction Processing) jsou označovány systémy transakční, jejich účelem je neustálé zaznamenávání provozních dat informačního systému. OLTP systémy jsou navrženy pro provádění malých a krátkých transakcí, které jsou však velmi časté. Často jsou tyto systémy využívány více aplikacemi a řada transakcí probíhá paralelně. Návrh databázové struktury OLTP systémů je zaměřen na snížení redundance ukládaných dat a snížení funkčních závislostí mezi daty. Data v OLTP systémech jsou ukládána v normalizované podobě. Databáze OLTP systémů potom obsahují velké množství tabulek s mnoha úrovněmi zanoření. Struktura databází OLTP je přizpůsobena častému vkládání nových dat, změnám v provozních datech a mazání těchto dat. OLTP systémy jsou zdrojem dat pro datové sklady. [1]

1.2 OLAP systémy

Zkratkou OLAP (On-Line Analytical Processing) je označována aktivita dotazování a prezentace textových a numerických dat datových skladů. Technologie uložení dat je většinou nerelační a je založena na vícerozměrných datových kostkách. Pro uložení dat OLAP systémy využívají multidimenzionální databázový model.[2]

Technologie uložení dat je optimalizovaná na snadné a rychlé provádění výběrových dotazů a analýz dat.

1.3 Datový sklad

Datový sklad je systém, který přenáší a sjednocuje data periodicky ze zdrojových systémů na dimenzionální nebo normalizované datové úložiště. Datový sklad obvykle potom slouží pro získávání výsledků dotazů pro nástroje business intelligence a jiné analytické procesy. Datový sklad je aktualizován a plněn periodicky v dávkách ze zdrojových systémů. K plnění datového skladu tedy nedochází okamžitě po každé změně v primárním datovém zdroji, ale pravidelně po uplynutí definované periody. [3]

V minimální verzi se systém každého datového skladu skládá ze dvou podsystémů. Prvním podsystémem je se systém ETL, sloužící k extrakci dat z primárních zdrojů, transformaci a pravidelnému uložení v cílovém úložišti, kterým nemusí být vždy datový sklad. Druhý podsystém tvoří obvykle dimenzionální datové úložiště. Datové úložiště data uchovává v jiném formátu než zdrojové OLTP systémy. [3]

Zatímco zdrojové transakční OLTP systémy jsou optimalizovány na stálé zaznamenávání provozních dat a jejich aktualizaci, datové sklady slouží k dlouhodobému uložení dat. Data do datových skladů se vkládají v dávkách a po vložení není účelem tato data měnit. Cílem datových skladů je uchovávat data ve formátu optimalizovaném pro poskytování výsledků na výběrové dotazy a pro analýzu dat, spíše než pro zaznamenávání a ukládání provozních dat a provádění jejich aktualizací.

Datové sklady mohou být postaveny na dimenzionálním nebo normalizovaném datovém úložišti. Dimenzionální datová úložiště využívají strukturu tabulek ve formě hvězdy nebo sněhové vločky. Databáze je potom tvořena datovými tabulkami označované jako tabulky faktů, na které jsou vázány popisné tabulky jednotlivých dimenzí. Výhodnou dimenzionálních datových struktur je snadnější provádění výběrových dotazů, neboť tabulky v případě uspořádání do hvězdy obsahují vždy pouze jednu úroveň zanoření. Nevýhodnou dimenzionálních datových úložišť je redundantní uložení dat. Redundantní uložení dat stěžuje případné provedení aktualizace, protože stejná data jsou uložena ve více tabulkách. [3]

Datové sklady mohou pro ukládání dat využívat také databáze v normalizované podobě. Normalizaci můžeme definovat jako proces odstranění redundance dat použitím normalizačních pravidel. Obvykle je požadováno dosažení třetí normální formy. Výhodnou normalizovaných datových úložišť je odstranění redundance dat. Normalizovaná datová úložiště však mají větší počet menších tabulek s více úrovněmi zanoření. Nevýhodou je

potom obtížnější sestavování výběrových dotazů neboť je potřeba provést více spojení mezi tabulkami. Provádění složitých výběrových dotazů nad normalizovanou databází je také časově složitější kvůli velkému množství spojení tabulek než nad vícerozměrnou databází. [3]

Bill Inmon definuje datový sklad jako podnikově strukturovaný depozitář subjektivě orientovaných, integrovaných časově proměnných historických dat použitých pro získávání informací a podporu rozhodování. [1]

Inmonova definice říká, že data v datovém skladu jsou subjektivě orientovaná. Data jsou ukládána v tabulkách podle předmětu zájmu. Data jsou integrovaná a uložena v jednotné formě bez ohledu na to, ze kterého zdroje dat pocházejí. V primárních zdrojích data mohou být uložena v různých formátech. Data jsou časově proměnná, jsou tedy do datového skladu ukládána v dávkách, kdy každá dávka reprezentuje určité časové období. K ukládaným datům je proto přidáván i časový údaj. Data uložena v datovém skladu jsou neměnná. Uložená data se nemění a neodstraňují, aktualizace datového skladu probíhá pouze přidáváním dat nových. [1]

Jinou definicí datového skladu je definice Ralpha Kimballa, který datový sklad definuje jako systém, který extrahuje, čistí, přizpůsobuje a odevzdává zdrojová data do vícerozměrného datového úložiště a pak podporuje a implementuje tvorbu dotazů a analýz pro podporu rozhodování. [3]

1.4 Datové trhy

Datové trhy jsou definované jako menší části datového skladu, které jsou vytvořené pro určitou organizační jednotku firmy. Datové trhy mohou být vytvořené jako samostatné menší datové sklady a plněny z hlavního datového skladu ETL procesy nebo mohou být součástí, podsystémem datového skladu. [1]

1.5 Import a export dat

Exportem dat můžeme definovat jako přesunutí dat z databázového systému do definovaného datového formátu. [1]

Importem dat se rozumí nahrání dat ze zdroje dat do databázového systému. Na rozdíl od ETL procesů, které jsou periodické a pravidelně se opakují, import dat je vždy operací jednorázovou. Import a export dat je obvykle prováděn v rámci instalace nového

informačního systému, migrace na jiný informační systém, přechodem mezi databázovými systémy, nebo při zavádění dat do datového skladu. [1]

1.6 ETL procesy

ETL proces je definován jako mechanismus získávání dat z provozních systémů podniku a jejich následné zpracování a poskytnutí aplikacím pro podporu rozhodování, kterými jsou decision support systémy, datové sklady a business intelligence. [4]

Zkratka ETL v sobě skrývá tři slova popisující proces přenosu dat ze zdrojového do cílového informačního systému. Extraction představuje postup získávání dat ze zdrojového systému. Transformation označuje postup úpravy a čištění načtených zdrojových dat do podoby vhodné pro jejich uložení v datovém skladu, kontrolu načítaných dat a přidání časových údajů k zpracovávaným záznamům. Loading pak popisuje postup nahrání získaných a upravených dat do datového skladu. ETL proces na rozdíl od migrace dat, která je prováděna jednorázově, je proces periodický a slouží k pravidelnému plnění dat do datového skladu.

Existuje více postupů, jak mohou být ETL procesy implementovány. V původním konceptu jsou data nejdříve extrahována z datového zdroje a uložena v dočasném úložišti, kde je provedena jejich transformace. Zpracovaná data jsou následně nahrávána do datového skladu. V případě malého objemu dat je fáze uložení v dočasném úložišti vynechávána a fáze transformace dat před uložením do datového skladu probíhá přímo v operační paměti, což je podstatně rychlejší, protože nedochází k zápisu na disk. [3]

Druhá alternativa bývá označována jako ELT proces, ve kterém jsou data nejdříve získávána z OLTP systémů, netransformovaná data jsou přímo ukládána do datového skladu, kde pak později probíhá jejich transformace. Původní verze ETL procesu klade větší nároky na kvalitu softwarového nástroje, který musí zajišťovat přenos a poskytovat potřebné transformační funkce. Naopak verze ELT klade větší nároky na datové úložiště, ve kterém pak probíhá transformační fáze celého procesu. [3]

V rámci datových skladů se ETL procesy uplatňují ve dvou vrstvách. První vrstva se označuje jako vrstva pořizování dat. V této vrstvě slouží ETL proces k přípravě a uchování dat. Tato vrstva slouží k naplnění datového skladu daty. [5]

Druhá vrstva je označovaná jako vrstva distribuce dat. Distribuce dat slouží k extrakci dat z oblasti uchovávání v datovém skladu a převod těchto dat do formátů konkrétních datových trhů. [5]

Existují dva odlišné postupy realizace ETL procesu. ETL proces může být postaven na vývoji samostatných programů nebo skriptů, které jsou vytvářeny vývojáři nebo na použití specializovaného softwarového produktu, ETL nástroje. Použití ETL nástroje přináší výhody, kterými jsou vyšší produktivita, flexibilita, výkon, otevřenost a podpora metadat. ETL nástroje nabízejí možnost návrhu ETL procesu v grafickém prostředí, které urychluje celou fázi návrhu. Vytvořený návrh je v případě změn možné snadno modifikovat. ETL nástroje jsou rovněž navrženy pro optimální využití softwarových a hardwarových prostředků a optimalizaci výkonu. Realizace ETL samostatným skriptem nebo programem přináší nevýhodu v nutnosti následné údržby programového kódu a obtížnější a náročnější realizaci změn v návrhu ETL procesu. Postup realizace ETL samostatnými skripty je však výhodnější a také rychlejší pro malá řešení, která nevyžadují pracnou údržbu [4, 6].

Rainardi dělí ETL procesy podle toho, kde jsou umístěny na ETL procesy běžící na zdrojových systémech, na ETL procesy umístěné na serveru datového skladu a na procesy běžící na samostatném ETL serveru. Výhodou samostatného ETL serveru je poskytnutí vyššího výkonu, protože fáze transformace dat nezatěžuje zdrojový ani cílový systém. Při umístění procesu ETL na server datového skladu jsou data přenášena, transformována a aktualizována v datovém skladu v období, kdy je datový sklad méně vytížen. V případě umístění ETL procesů ve zdrojových systémech jsou data ze zdrojových systémů vynášena pomocí triggerů při změnách ve zdrojových systémech. Toto řešení však zpomaluje zdrojový systém a je tak vhodné pouze v případech, kdy je nezbytně nutné mít data v datovém skladu aktuální. [3]

1.7 Fáze ETL procesu

Jak vyplývá ze zkratky ETL procesu, přenos dat probíhá ve třech fázích extrakce dat ze zdrojového systému, transformace dat do formátu vhodného pro cílový systém a fáze nahrání transformovaných dat do databáze cílového systému.

1.7.1 Extrakce

Fáze extrakce v sobě zahrnuje výběr potřebných dat ze zdrojových systémů. Všechna data obsažená ve zdrojovém systému však nemusí být potřebná pro další ukládání v datovém skladu, protože tato data nejsou požadována pro následující analýzu. Cílem fáze extrakce je tedy přesně určit podmnožinu zdrojových dat pro další zpracování v ETL procesu a pro uložení v datovém skladu. [6]

V konkrétním případě sběru dat inventarizace lesů, jsou tato data sbírána prostřednictvím programu Field-Map Data Collector. Tento program do souboru databáze Access ukládá, kromě sbíraných dat také své systémové informace. Pro další analýzu jsou však požadována pouze měřená data, jejichž sběr a postup měření je popsán v metodikách sběru. Je tedy potřeba zvážit, zda je s měřenými daty také nezbytně nutné ukládat i systémová data externí aplikace pro sběr dat.

Extrakce dat ze zdrojového systému by neměla způsobit zatížení a zpomalení zdrojového systému. Je proto třeba, aby množství přenášených dat bylo co nejmenší a aby proces extrakce probíhal co nejméně často a aby byl rychlý. Je také třeba, aby fáze extrakce nevyžadovala žádné změny ve zdrojovém systému. [3]

Extrakce dat může být prováděna ETL nástrojem, který pravidelně spouští výběrové dotazy nad zdrojovými systémy, které slouží k získání přenášených dat. Data pro datový sklad mohou být také získávána pomocí triggerů na straně zdrojových systémů. Triggery tabulek zdrojového systému jsou spuštěny při změnách obsahu tabulek smazáním, změnou nebo vložením nových záznamů. Změněné záznamy jsou potom ukládány do tabulek pro import do datového skladu. Data mohou být také získávána pravidelným spuštěním uložené procedury na straně zdrojového systému. Extrahovaná data je možné také načítat ze souborů databázového logu, který uchovává historii změn v datových tabulkách. [3]

Zdrojová data mohou pocházet z více systémů, které jsou různorodé. Data mohou být uložena v různých formátech a v rozdílné kvalitě.

1.7.2 Transformace

Transformace je proces převodu dat načtených ze zdrojového systému do struktury, která je vhodná pro uložení v datovém skladu. Transformace mohou být provedeny na úrovni záznamů nebo na úrovni hodnot v záznamech. [7]

Transformace je možné rozdělit na transformace struktury, obsahu a funkční transformace. Strukturní transformace mění strukturu zdrojových sloupců na strukturu vyhovující struktuře sloupců v cílové databázi. Transformace obsahu umožňuje měnit hodnoty v záznamech. Změny mohou být provedeny podle stanoveného algoritmu nebo transformační tabulky. Funkční transformace ukládají v tabulkách nové hodnoty, které byly dopočítány ze zdrojových dat. Funkčními transformacemi jsou výpočty hodnot agregačních funkcí, kterými jsou například výpočty průměrných hodnot, získání maximálních a minimálních hodnot. Funkčními transformacemi jsou také dopočítání nových hodnot z více hodnot vstupních. [7]

1.7.3 Nahrání dat

Nahrání dat je poslední fází přenosu dat ze zdrojového systému do datového skladu. Existují čtyři možné postupy nahrání dat. Při prostém nahrání dat nová data nahrazují existující data v cílové tabulce, která jsou před nahráním dat z tabulky odstraněna, nebo je vytvořena nová cílová tabulka. Druhým způsobem je připojení dat, kdy stará data v cílové tabulce jsou zachována a nová data jsou přidána do existující cílové datové tabulky. Při konstruktivním spojení jsou nová data do cílové tabulky nahrána a existující nahrazované datové záznamy jsou aktualizovány nastavením času konce platnosti. Při destruktivním spojení jsou existující záznamy v datové tabulce přepsány novými záznamy, které je nahrazují. [7]

1.8 Problémy ETL procesů

Existuje řada problémů, které je nutné řešit při návrhu ETL procesu. Prvním problémem je problém nejednoznačných číselníků. Pokud jsou data získávána z více systémů, pak každý systém může používat různé číselníky. Jedna kategorie v číselníku jednoho systému může mít jiný kód než tatož kategorie v číselníku druhého systému. Při uložení těchto dat v datovém skladu pak jedné kategorii budou přiřazeny dva kódy v číselníku. Tato nepřesnost způsobuje problémy při analýze dat. Dalším problémem v číselnících může být přiřazení téhož kódu dvěma kategoriím. [6]

V případě dat pozemního šetření inventarizace lesů byly číselníky navrženy před zahájením terénního sběru dat. Všechny soubory, do kterých jsou data měření ukládána, používají tytéž sjednocené číselníky. V průběhu sběru dat však mohou vznikat

potřeby na úpravy číselníků. Touto potřebou může být například příchod nového člena měřicí skupiny, pro kterého musí být založena nová kategorie v číselníku.

Používání číselníku zabraňuje vzniku problému s nejednoznačností dat. V datových tabulkách jsou místo textových řetězců používány číselné kódy kategorií, které jednoznačně identifikují kategorii v číselníku, kde je uveden její popis.

Dalším možným problémem je nedodržení referenční integrity dat. Primárním systémem, který je zdrojem dat, nemusí být databázový systém a není zaručeno, že primární systém bude mít definovány relační vazby mezi tabulkami a využívat možností databázových systémů. V těchto zdrojích dat jsou data uložena bez zajištění jejich referenční integrity nebo je referenční integrita pouze vynucována a ověřována softwarovou aplikací, která slouží k pořízení těchto dat. [6]

Měření inventarizace lesů jsou ukládána v souborech databáze Access. Databáze Access umožňuje samozřejmě definovat relační vazby mezi uloženými tabulkami. Aplikace sloužící ke sběru dat pozemního šetření inventarizace však tuto možnost nevyužívá. Informace o existujících tabulkách a vazbách mezi tabulkami si tato aplikace ukládá do svých systémových tabulek, které jsou součástí datového souboru spolu s tabulkami měřených dat. Referenční integrita dat je pak vynucována aplikační vrstvou a není ve vrstvě datové. Pokud dojde ke smazání nebo editaci dat přímo bez použití aplikace pro sběr dat, hrozí riziko porušení referenční integrity.

Často v datech může vzniknout problém s duplicitou dat. Pokud data pocházejí z více zdrojových systémů, pak tyto systémy mohou obsahovat popis stejného objektu. V těchto případech je nutné rozhodnout, které hodnoty budou použity. [6]

Při sjednocení dat z více zdrojových systémů je možné také narazit na problém s různými použitými formáty pro čísla, textové řetězce a časové údaje. Některé zdrojové systémy zapisují desetinná čísla s oddělovačem desetinné čárky, jiné používají oddělovač desetinou tečku.

Problémy ve fázi transformace je možné rozdělit na problémy vyskytující se na úrovni schématu, na úrovni záznamu a na úrovni jednotlivých datových hodnot. [8]

1.9 Kontrola kvality dat

Data, která jsou získávána sběrem a měřením a následně uložena v datových skladech, jsou zpravidla pořizována se záměrem zpracování konkrétní analýzy, dosažení předem definovaného cíle a získání požadované informace. Aby informace získané zpracováním dat uložených v datovém skladu byly spolehlivé, je nutné zabránit chybným datům, aby se dostávala do datového skladu.

V datech importovaných do datového skladu se mohou vyskytovat chyby, které je možné rozdělit do několika skupin [3]:

- Nesprávná data tvoří skupinu dat, kde požadovaná hodnota atributu je změřena s chybou a tato hodnota neodpovídá skutečnosti.
- Chybějící data jsou taková, kde hodnota atributu popisující danou entitu chybí, v případě, že má být vyplněna nebo dokonce chybí celý záznam popisující entitu.
- Nepřesná data. Data jsou obvykle pořizována s předem definovaným záměrem. Přesnost, s jakou jsou dané údaje měřené, musí umožňovat dosažení cíle, pro který data byla sbírána.
- Zastaralá data. Hodnoty měřených atributů mohou být závislé na čase a jejich velikost se s časem měnit.
- Nekonzistentní data. Mezi jednotlivými zjišťovanými atributy mohou existovat logické vazby. Získaná hodnota jednoho atributu by pak neměla být ve vzájemném rozporu s jinou hodnotou jiného atributu.

Na začátku každého projektu datového skladu je nezbytně nutné definovat množinu pravidel, která přesně určují, co jsou chybná data. Definování pravidel pro kontrolu dat a vymezení intervalu, ve kterém se nacházejí správné údaje o měřené veličině, není často snadné. Například pokud měřenou veličinou bude výška stromu, pak o záporné hodnotě můžeme s jistotou říct, že se jedná o chybu a požadovat opravu záznamu před jeho importem do datového skladu. Určení horní hranice výšky však představuje problém, protože není možné tuto hranici definovat přesně. V těchto případech bývá určen interval, ve kterém daná hodnota ještě není považována za chybnou, je importována do datového skladu, ale o importu je vytvořen a reportován záznam jako varování.

Proces kontroly kvality dat zajišťuje, aby data v datovém skladu byla správná a úplná. Představuje činnost kontrolování dat, reportování chyb a jejich opravu. [3]

Data jsou kontrolována pomocí datového firewallu, který bývá implementován pomocí uložených procedur. Datový firewall tato data kontroluje, zda vyhovují pravidlům definujícím bezchybná data. Tato pravidla jsou uložena v databázi metadat. Pokud data pravidla splňují, jsou uložena v cílové databázi datového skladu. V opačném případě jsou data uložena v databázi chybných dat, která obsahuje stejné tabulky jako cílová databáze. Výskyt chyb je pravidelně reportován a je požadována oprava záznamů ve zdrojových systémech. [3]

2 PŘEHLED ETL NÁSTOJŮ

Tato část práce se zaměřuje na popis použití několika ETL nástrojů. Vzhledem k nezměrnému množství nástrojů použitelných při migracích dat do datových skladů není možné uvést kompletní výčet všech existujících řešení. Stejně tak je velmi obtížné přesně definovat hranici, která by určovala, který softwarový nástroj je nástrojem ETL. Některé programy jsou zaměřeny spíše na správu databázových systémů nebo editaci SQL dotazů, tyto programy však mají řadu funkcí, které umožňují migraci dat a jejich transformaci. Řada databázových systémů má rovněž svá vlastní řešení pro import a export dat.

V následujícím výčtu bude tedy dána přednost spíše nástrojům poskytovaných jako open source, které jsou volně dostupné. Řada ETL nástrojů existuje také ve dvou verzích, ve verzi komerční, ke které je poskytována podpora a ve verzi volně dostupné. Požadováno je, aby tyto nástroje umožňovali migraci dat mezi soubory databáze Access, ve kterých jsou uložena sbíraná data inventarizace lesů, do databáze PostgreSQL, která slouží jako datový sklad inventarizace lesů.

Schiller dělí ETL nástroje do dvou skupin a to na ETL nástroje první a druhé generace. Jako nástroje první generace označuje softwarové produkty, které na základě navrženého transformačního předpisu generují kód, který je spuštěn na zdrojové platformě a na nástroje druhé generace, které realizují ETL proces podle objektově definovaného předpisu uloženého v katalogu metadat. [4]

2.1 Clover ETL

Technologie Clover je založena na třech softwarových aplikacích. Aplikace *Clover ETL Designer* je grafické prostředí určené k návrhu ETL procesu ve formě transformačního grafu. *Clover ETL Engine* je výkonnou částí aplikace, která provádí transformaci dat podle vytvořeného návrhu v transformačním grafu. *Clover ETL Server* slouží k automatizaci, plánování a centralizované správě mnoha transformačních úloh. [9]

Clover ETL je komerční aplikací, která je dostupná ve třech edicích. Komerční označená jako *Desktop Edition*, zkušební časově omezená *Desktop Trial Edition* a volně dostupná *Community Edition*.

Součástí *Community Edition* je open source transformační engine a omezené grafické vývojové prostředí *Clover ETL Designer*. Vývojové prostředí je omezeno množstvím

dostupných typů komponent na 26 a možností použití maximálně 20 komponent v návrhu jednoho transformačního grafu.

Clover ETL je aplikací založenou na programovacím jazyce Java a je možné ji integrovat s vývojovým prostředím Eclipse. Clover ETL je aplikací multiplatformní. Existují instalace pro operační systémy Windows, Mac OS a Linux.

Jednotlivé transformační úkoly jsou ukládány ve formě projektů do adresáře pracovního prostředí (*workspace*). Do pracovního prostředí je možné ukládat více projektů. Každý projekt má definovanou adresářovou strukturu, kdy jednotlivé adresáře slouží k ukládání souborů zdrojových dat, dočasných souborů, souborů výstupních dat a souborů popisujících ETL proces. Soubory definující ETL proces nesou informace o připojeních ke zdrojovým a cílovým databázím, popis transformací, definice transformačního grafu, číselníky a definice metadat. Všechny soubory definující ETL proces aplikace ukládá ve formátu xml souborů. ETL Designer zobrazuje strukturu otevřeného projektu na panelu Navigator.

2.1.1 Komponenty a sestavení transformačního grafu

Návrh ETL procesu probíhá vytvořením transformačního grafu, který se skládá z jednotlivých komponent propojených liniemi představujícími datové toky mezi jednotlivými komponentami. Návrhář vybírá potřebné komponenty z panelu Paleta a tyto komponenty umísťuje na plochu grafu. Pro jednotlivé komponenty poté definuje jejich vlastnosti. Komponenty mají podle svého typu vstupy a výstupy. Přenos dat mezi komponentami je v grafu zakreslen liniemi, které propojují výstup jedné komponenty na vstup druhé. Každá linie je definována pomocí metadat, která popisují formát dat předávaných mezi komponentami.

Množství dostupných komponent a jejich skupin je různé a omezené podle edice aplikace. V případě *Community Edition* jsou komponenty na panelu Paleta rozděleny do pěti skupin. Skupina *Readers* obsahuje komponenty určené ke čtení vstupních dat z různých datových zdrojů. Komponentami této skupiny začíná datový tok v transformačním grafu. Skupina *Writers* naopak zahrnuje komponenty určené k zápisu výsledných dat do cílových datových úložišť. Skupina *Transformers* obsahuje komponenty pro modifikaci dat na cestě mezi zdrojem dat a cílovým úložištěm. Skupina *Joiners* obsahuje komponenty pro spojení datových toků podle hodnot definovaného klíče. Poslední skupinou je skupina *Others* s ostatními komponentami s různorodým účelem.

Skupina *Readers* obsahuje následující komponenty [9]:

- *DBFDataReader* je komponenta určená ke čtení souborů databáze dBase.
- *DBInputTable* komponenta umožňuje čtení dat databázové tabulky z databáze, ke které existuje připojení pomocí rozhraní JDBC.
- *UniversalDataReader* je komponenta určená ke čtení dat ze zdrojových textových souborů s definovanými oddělovači datových sloupců nebo se sloupci s pevnou šířkou, které mohou být uloženy také v komprimovaných zip archivech.
- *XLSDataReader* je komponenta, kterou je možné číst zdrojová data ze souborů listů aplikace Microsoft Excel.
- *XMLReader* je komponenta určená pro čtení dat z xml souborů.

Skupina *Writers* obsahuje komponenty [9]:

- *DBOutputTable* komponenta slouží k zápisu výsledných dat do cílové databáze určené připojením prostřednictvím JDBC.
- *XLSDataWriter* umožňuje zápis do formátu souborů xls a xlsx aplikace Microsoft Excel.
- *UniversalDataWriter* umožňuje zápis do textových souborů s definovaným oddělovačem datových sloupců nebo s pevnou šířkou sloupců.
- *Trash* komponenta umožňuje zakončení datového toku zrušením jeho výsledku bez uložení.

Skupina *Transformers* obsahuje komponenty [9]:

- *Aggregate* komponenta z dat na svém vstupním portu vytvoří skupiny podle definovaného agregačního klíče a provede výpočty hodnot agregačních funkcí pro tyto skupiny. Výsledek agregačních funkcí předává na svém výstupním portu.
- *Concatenate* komponenta sloučí data ze všech svých vstupních portů, která mají stejnou strukturu definovanou metadaty a vloží je na výstup podle pořadí na vstupních portech.
- *Dedup* komponenta slouží k deduplikaci vstupních dat. Na základě deduplikačního klíče jsou identifikované stejné skupiny dat, komponenta na výstupu vrátí definovaný počet záznamů z každé skupiny určené deduplikačním klíčem.
- *ExtFilter* komponenta umožňuje výběr datových řádků na vstupním portu podle definované výběrové podmínky.
- *ExtSort* komponenta slouží k seřazení záznamů podle definovaného klíče.

- *Merge* komponenta je určena ke sloučení více seřazených vstupů, které vrátí na výstup seřazené podle téhož klíče.
- *Reformat* komponenta umožní přijímat data na vstupu a definovat mapování vstupních sloupců na výstupní. Při přenosu dat mezi vstupem a výstupem umožní data transformovat pomocí definovaných funkcí.
- *SimpleCopy* komponenta datové řádky přijaté na svém vstupu kopíruje na všechny své výstupní porty.
- *SimpleGather* komponenta datové řádky přijaté na svých vstupech vkládá do jediného výstupu. Všechny vstupní a výstupní porty musí mít stejnou strukturu popsanou metadaty.

Skupina *Joiners* obsahuje pouze jedinou komponentu *ExtHashJoin*. Komponenta přijímá na prvním vstupu datové řádky, které spojuje s datovými řádky ostatních vstupů podle stejných hodnot v sloupcích určených jako klíč [9].

Skupina *Others* tvoří komponenty [9]:

- *DBExecute* je komponenta, která umožní spuštění definovaného příkazu jazyka SQL v databázi definované připojením pomocí rozhraní JDBC.
- *SystemExecute* je komponenta, která umožňuje spustit příkaz, který je vykonán operačním systémem.

2.1.2 Definice metadat

Všechny transformace a operace s daty jsou prováděny pouze v rámci komponent. Komponenty propojené liniemi si mezi sebou předávají data v různém stádiu zpracování. Při přenosu dat mezi komponentami se data nemění. Každá spojovací linie musí mít definován formát dat, v jakém jsou data mezi komponentami transformačního grafu předávána, a který je popsán metadaty. Metadata určují, jaké atributy budou mít předávané záznamy a jakých datových typů budou tyto atributy. V rámci projektu je tedy nejdříve definováno několik skupin metadat pomocí formuláře *Edit metadata* (Obr. 1), tyto skupiny jsou pojmenovány a uloženy. Každé propojovací linii v transformačním grafu je pak přiřazena zvolená skupina metadat.

Edit metadata

Field: diameter

#	Name	Type	Delimiter	Label
1	Record: tree_metadata	delimited	;	tree_data_01.csv
1	id_plot	integer	;	id_plot
2	id_tree	integer	;	id_tree
3	species	integer	;	species
4	diameter	integer	;	diameter
5	height	decimal	;	height
6	age	integer	\n	age

Property Value

Name	diameter
Label	diameter
Type	integer
Container type	
Delimiter	;

Selected field is valid

id_plot	id_tree	species	diameter	height	age	Input
42050	1	6	254	21.6	65	
42050	2	6	211	16	65	
42050	3	6	235	19.2	65	
42050	5	6	188	13.2	65	
42050	6	6	243	21.4	65	
42050	7	6	230	18.7	65	
42050	8	6	278	18.6	65	
42050	9	6	274	18.7	65	

Lines of input file for preview: 10

```
id_plot;id_tree;species;diameter;height;age
42050;1;6;254;21.6;65
42050;2;6;211;16.0;65
```

Attached preview: \$(DATAIN_DIR)/tree_data_01.csv

ISO-8859-1 Browse... Remove

Lock OK Cancel

Obr. 1. Clover ETL – formulář definice metadat

Pokud v rámci komponenty dochází ke změně struktury dat, vstupní data jsou popsány jinými metadaty než data výstupní, je nutné v rámci komponenty definovat mapování jednotlivých vstupních atributů na výstupní. Toto je možné provést v grafickém návrhu prostým přiřazením vstupních atributů na výstupní nebo transformaci popsat pomocí jazyka CTL (Clover Transformation Language). Hodnota výstupu může být pak dopočtena z hodnot vstupních.

2.1.3 Definice připojení

Komponenty pracující s databázemi vyžadují definování připojení k databázi. Pro připojení k databázím aplikace Clover využívá JDBC ovladačů. V rámci jednoho projektu je možné definovat více připojení, tato připojení pojmenovat a používat je v komponentách transformačního grafu. Definice nového připojení je provedena na formuláři *Database connection* (Obr. 2). Po zvolení JDBC ovladače aplikace nabídne šablonu URL, do které uživatel doplní IP adresu databázového serveru, TCP port a jméno připojované databáze.

Database connection
Define database connection

Basic | Advanced

Connection: pg_nfi_data_connection (id:JDBC0) [Load from file]

Name: pg_nfi_data_connection

User: postgres

Password: *****

URL: jdbc:postgresql://127.0.0.1:5432/DS_NIL

JNDI:

JDBC specific: PostgreSQL

Available drivers:

- Derby
- Firebird
- Generic ODBC
- Microsoft Access
- Microsoft SQL Server
- MySQL
- Oracle
- PostgreSQL**
- SQLite
- Sybase

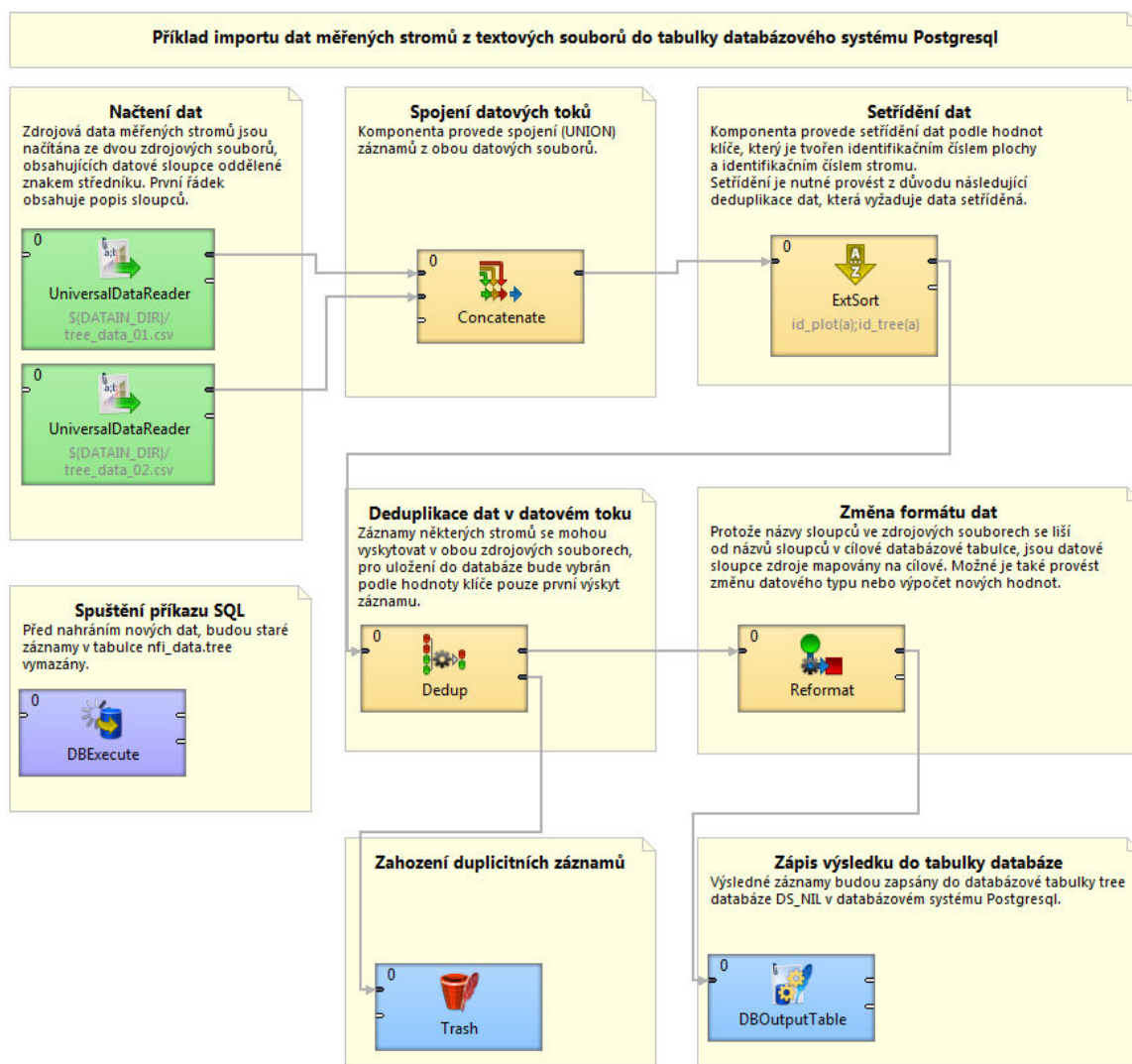
Validate connection

Lock OK Cancel

Obr. 2. Clover ETL – formulář definice připojení

2.1.4 Příklad importu dat aplikací Clover ETL

Následující příklad popsaný transformačním grafem (Obr. 3) demonstruje použití aplikace Clover při importu dat z textových souborů do tabulky databázového systému PostgreSQL. Zdroj dat představují textové soubory obsahující měřená data stromů. Řádky v textových souborech představují údaje získané měřením stromu, kterými jsou výška, výčetní tloušťka kmene, věk, druh dřeviny a identifikace stromu v rámci měřené plochy. Data jsou oddělena znakem středníku. První řádek obsahuje názvy atributů a bude při importu zahozen.



Obr. 3. Clover ETL - transformační graf pro import dat

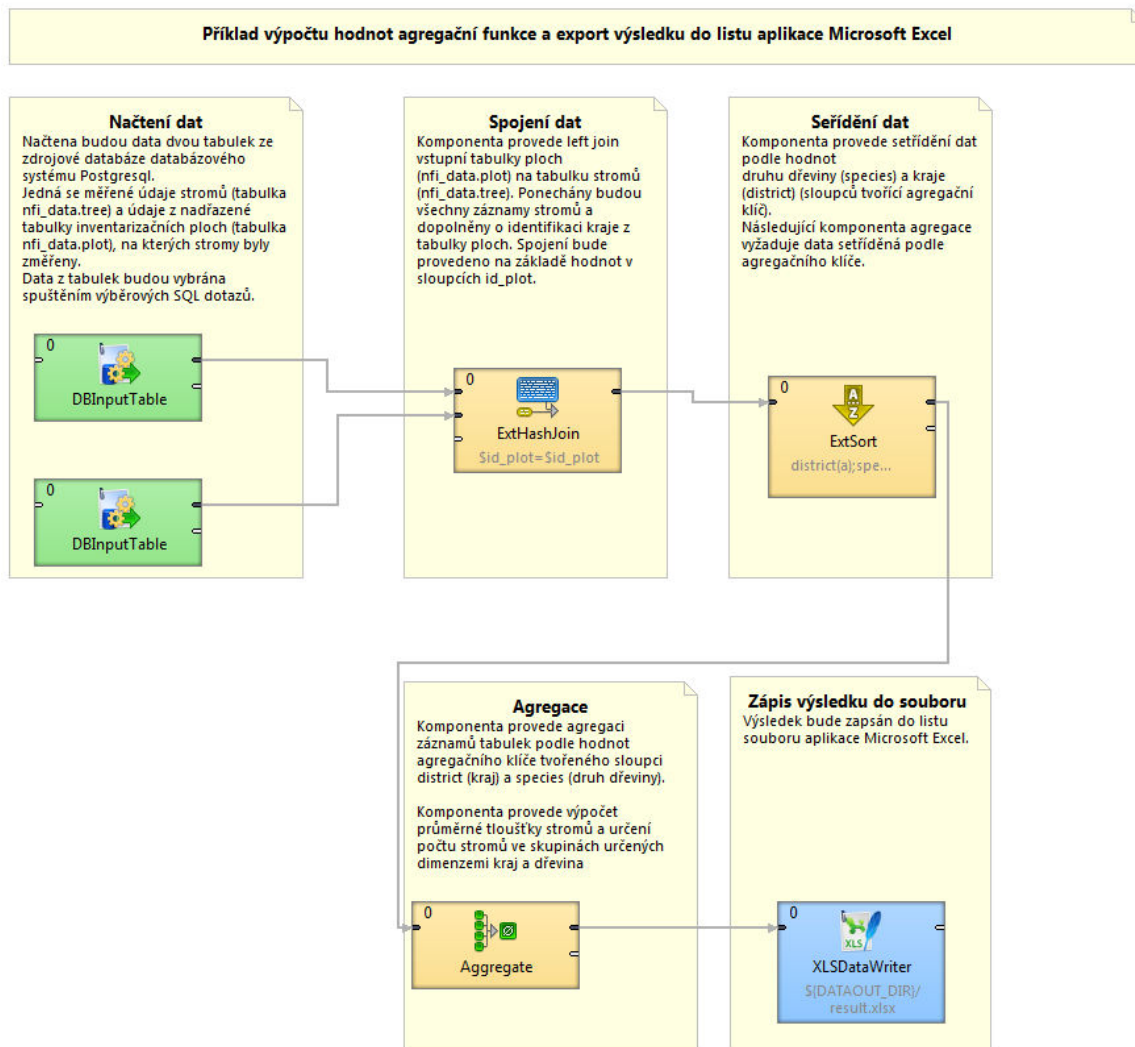
Při spuštění transformačního grafu jsou nejdříve načtena zdrojová data z textových souborů v komponentách *UniversalDataReader*, jejichž vlastnosti určují umístění zdrojového souboru na disku a v jakém formátu jsou data v textovém souboru zapsána. Načtená data jsou předána komponentě *Concatenate*, která spojí oba datové toky do jednoho. Datový tok dále pokračuje komponentou *ExtSort*. Komponenta obsahuje definici klíče, podle kterého mají být data vzestupně seříděna. Seřídění podle klíče je nutné pro další zpracování, protože následující komponenta transformačního grafu vyžaduje data seříděná. Protože data načítáme z více souborů, může dojít k tomu, že některé datové řádky obou souborů popisují tentýž strom určený identifikačními čísly plochy a stromu. Tato duplicitní data není možné uložit do databázové tabulky, protože hodnoty primárního klíče musí být unikátní. Následující komponenta *Dedup* provede deduplikaci přijatých dat. Podle hodnoty deduplikačního klíče je ponechán pouze první

výskyt stromu určeného hodnotou klíče. Ostatní záznamy umístí komponenta *Dedup* na druhý výstupní port a tyto jsou poté zahozeny umístěním do komponenty *Trash*. Protože názvy datových sloupců jsou různé ve zdrojových souborech a v cílové databázové tabulce je následující komponentou v datovém toku komponenta *Reformat*. V dialogu komponenty *Reformat* bylo určeno mapování sloupců ze vstupního portu na port výstupní. Data na vstupním portu jsou tedy popsána jinými metadaty než data na portu výstupním. Poslední komponentou v datovém toku je komponenta *DBOutputTable* v jejích vlastnostech bylo specifikováno připojení k databázi PostgreSQL a zvolena cílová tabulka. Komponenta provede zápis záznamů přečtených na svém vstupním portu do cílové tabulky. Komponenta stojící mimo datový tok *DBExecute* obsahuje definici SQL příkazu DELETE, který zajišťuje, že před spuštěním transformačního grafu budou staré záznamy před nahráním nových z cílové tabulky vymazány.

2.1.5 Příklad exportu dat aplikací Clover ETL

Příklad popsaný transformačním grafem (Obr. 4) popisuje možný postup výpočtu agregovaných statistických hodnot (průměr, medián, směrodatná odchylka, počet stromů) z dat stromů uložených v databázových tabulkách, pro skupiny popsané dimenzemi dřevina a kraj, ve které leží plocha, na níž byly stromy měřeny. Datový tok v transformačním grafu začíná načtením dat v komponentách *DBInputTable*. Komponenty *DBInputTable* vyberou záznamy z tabulky inventarizačních ploch a tabulky stromů podle výběrového dotazu definovaného ve vlastnostech komponent nad připojením k databázovému systému PostgreSQL a výsledky dotazů vrátí na svém výstupním portu. Následující komponenta *ExtHashJoin* provede spojení dat obou tabulek podle stejných hodnot spojovacího klíče, kterým je identifikační číslo plochy. Na první vstupní port označovaný jako *master* byla přivedena data stromů a tato data byla doplněna z dat ploch načtených na portu *slave* o atribut kraje. Typ spojení byl nastaven na *left outer join*. Protože data na vstupu a výstupu jsou popsána jinými metadaty je nutné v rámci komponenty *ExtHashJoin* definovat mapování vstupních datových sloupců na výstupní. Doplněná data byla předána na výstupním portu následující komponentě *ExtSort*. Před provedením agregace je vyžadováno, aby data byla seříděna podle hodnot agregačního klíče tvořeného v tomto případě atributy kraje a druhu dřeviny. Předposlední komponenta *Aggregate* provede sloučení dat do skupin a na data skupin aplikuje agregační funkci. Dialog *Aggregation Mapping* aplikace Clover nabízí velké množství agregačních funkcí. V příkladu byla použita funkce *avg* pro výpočet aritmetického průměru tloušťek

stromů a funkce *count* pro stanovení počtu stromů ve skupinách. Výsledky agregačních funkcí jsou mapovány na výstup, který je opět popsán svými metadaty. Výstup je předán komponentě *XLSDataWriter*, která v souboru formátu Microsoft Excel, který je určen cestou k umístění na disku, vytvoří nový list a zapíše výsledky výpočtu.



Obr. 4. Clover ETL - transformační graf pro export dat

2.2 Apatar

Apatar je open source aplikací sloužící k integraci dat. Apatar byl vytvořen v programovacím jazyce J2EE ve vývojovém prostředí Eclipse. Apatar je aplikací multiplatformní, kterou je možné spustit v operačních systémech Windows, Linux a MacOS. [10]

Návrh úkolu přenosu dat mezi zdrojovým a cílovým systémem probíhá v grafickém prostředí označovaném jako datová mapa (*datamap*). Datová mapa tvoří převážnou část pracovní plochy hlavního formuláře aplikace. Při sestavování datové mapy se používají jednotlivé komponenty, které je možné na datovou mapu umístit jejich výběrem a přetažením ze seznamu v levé části hlavního formuláře.

Seznam obsahuje tři typy komponent *Connectors*, *Operations* a *Data Quality Services*. Komponenty typu *Connectors* představují datové zdroje, ke kterým je možné nastavit připojení. Prostřednictvím *Connectors* se také určuje cílová databáze nebo soubor pro zápis dat. Komponenty typu *Operations* pak obsahují operace, kterými je možné upravovat datový tok mezi zdrojovým a cílovým připojením a umožňují přenášená dat transformovat. Komponenty *Data Quality Services* určující přístup ke službám pro validaci dat, například telefonních čísel nebo e-mailových adres. [10]

2.2.1 Datové zdroje

Apatar podporuje velkou řadu datových zdrojů a formátů vstupních dat. Vstupním zdrojem dat mohou být různé aplikace, webové služby a protokoly. Podporovaným zdrojem dat je také velká řada databázových systémů a souborových databází. Apatar umožňuje připojení do databázových systémů IBM DB2, DBase, Firebird, Microsoft Access, Microsoft SQL Server, MySQL, OpenEdge, Oracle, PostgreSQL, Sybase, Vertica. Pokud zdrojová databáze nemá vlastní konektor, je možné se k této databázi připojit prostřednictvím konektoru OdbcGeneric. Předpokladem pro použití toto připojení je nainstalovaný ODBC ovladač pro příslušný databázový systém. Apatar umožňuje data importovat a exportovat z formátovaných textových souborů, souborů csv a souborů aplikace Microsoft Excel a také souborů XML. [10]

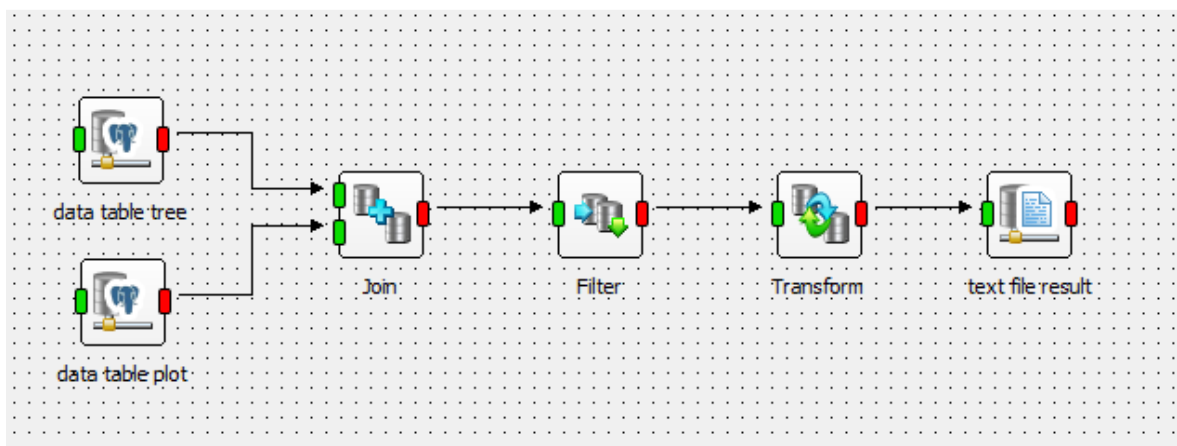
2.2.2 Operace

Komponenty *Operations* jsou určeny k práci s daty načtenými z datových zdrojů a jejich transformaci do cíle dat. Apatar nabízí tyto možné operace s daty:

- Operace *Aggregate* je určena k sloučení dat z více datových zdrojů.
- Operace *Distinct* vyřazuje stejné záznamy ze zdroje dat.
- Operace *Execute* umožňuje znovu spuštění integrační úlohy nebo spuštění definované externí aplikace.
- Operace *Filter* dovolí definovat podmínku pro výběr dat z datového zdroje.
- Operace *Join* slouží ke spojení dat různých tabulek a datových zdrojů podle propojovací podmínky.
- Operace *Transform* je určena k změně dat a jejich transformaci mezi zdrojem a cílem. Tato komponenta umožňuje definovat mapování sloupců tabulek datového zdroje na sloupce cílové tabulky. Komponenta nabízí velké množství funkcí, které je možné aplikovat na přenášená data jednotlivých sloupců.
- Operace *Update* je určena k synchronizaci dvou datových zdrojů.
- Operace *Validate* umožňuje definovat podmínku na základě, které je datový tok rozdělen do dvou větví, z nichž jedna obsahuje data vyhovující podmínce a druhá data, která podmínku nesplňují. [10]

2.2.3 Příklad exportu dat v aplikaci Apatar

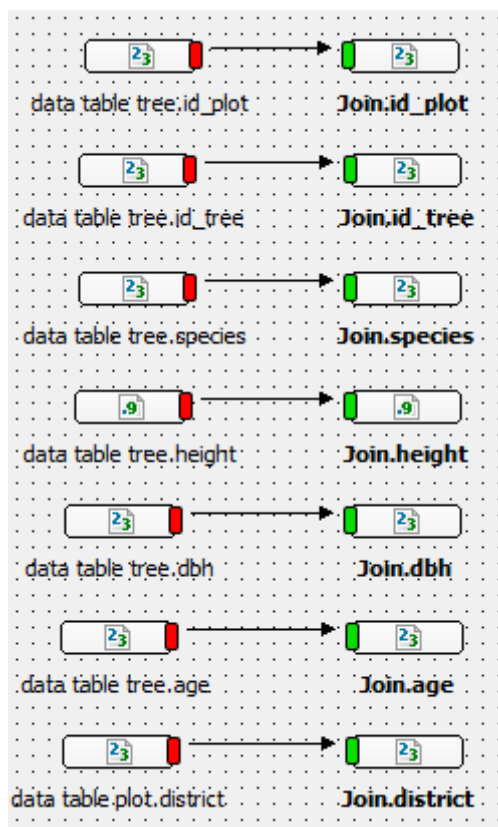
Datová mapa příkladu (Obr. 5) začíná načtením vstupních dat z tabulek stromů a ploch databázového systému PostgreSQL v komponentách PostgreSQL Connector.



Obr. 5. Apatar - datová mapa

V komponentách jsou nastaveny parametry připojení k databázovému systému a určen výběrový SQL dotaz. Výsledek dotazu komponenty vrací na svém výstupu. V následujícím kroku jsou datové toky spojeny operací Join. V komponentě operace *Join* je nutné definovat na základě, kterých datových sloupců bude provedeno spojení, typ spojení a určit

mapování sloupců mezi vstupním a výstupním datovým tokem. Návrh mapování sloupců je proveden v grafickém prostředí (Obr. 6), ve kterém jsou nabídnuty vstupní a výstupní datové sloupce, které je nutné pouze propojit. Do grafického návrhu mapování je možné umístit také komponenty transformačních funkcí a na výstup předat data upravená. V dalším kroku datové záznamy procházejí přes komponentu operace *Filter*, ve které je nastavena omezující podmínka. Operace *Filter* dále propouští pouze záznamy stromů, která mají změřenou výšku. Před zápisem do výsledného souboru jsou data upravená do formátu výsledného souboru v komponentě operace *Transform*. Podobně jako operace *Join*, komponenta *Transform* obsahuje formulář pro definici mapování vstupních datových sloupců na výstupní a aplikace transformačních funkcí. Datový tok končí v komponentě *TextFile Connector*, která určuje soubor a formát pro zápis výsledku.



Obr. 6. Apatar - definice mapování sloupců

2.3 Scriptella

Scriptella je volně dostupným ETL nástrojem založeným na technologii Java. Scriptella podporuje připojení k mnoha datovým zdrojům prostřednictvím rozhraní JDBC. Aplikace umožňuje současné připojení k více datovým zdrojům, které je možné použít jako zdroj nebo cíl pro nahrání dat. Data je možné při přenosu transformovat. Pro výběr dat ze zdrojového systému a transformaci dat používá skripty v jazyce SQL. Scriptella je aplikací multiplatformní a má dostupné instalace pro operační systémy Windows a Linux. Kromě migrací dat je aplikaci možné také používat k ukládání SQL skriptů definujících strukturu databáze. V uložených SQL skriptech umožňuje části kódu nahradit parametry a tyto parametry definovat v samostatných souborech. Tímto postupem je například možné nahradit v SQL kódu názvy datových typů atributů tabulek, jejichž definice jsou rozdílné v různých databázových systémech. Scriptella tímto umožňuje vytvoření SQL kódu definic databázových objektů, který je nezávislý na cílovém databázovém systému. [11]

Aplikace Scriptella nemá žádné grafické uživatelské rozhraní a práce s aplikací probíhá výhradně na příkazovém řádku operačního systému. Spuštění aplikace probíhá spuštěním souboru *scriptella.jar*, kdy v parametrech je spuštěnému procesu předána cesta k definičnímu souboru popisujícím požadovanou migraci dat.

2.3.1 Definiční soubor aplikace Scriptella

Jako definiční soubor využívá aplikace soubor formátu xml s předepsanou strukturou. Struktura souboru je definovaná dtd souborem. Definiční xml soubor je složený z jednotlivých elementů, jejichž obsah definuje celý přenos dat.

Element *<etl>* je kořenovým elementem celého xml souboru.

Element *<connection>* je určen k definici připojení ke zdrojovým i cílovým databázovým systémům. V rámci dokumentu je možné vytvořit více připojení a tyto připojení pojmenovat přiřazením hodnoty id atributu. V attributech elementu *<connection>* je nutné určit použitý JDBC ovladač pro připojovaný databázový systém a definovat parametry připojovacího řetězce.

Element *<properties>* umožňuje definovat proměnné, které je možné používat v celém definičním souboru.

Element `<query>` obsahuje výběrový dotaz v jazyce SQL, který je spuštěn nad připojením definovaným v atributu `connection-id`. Syntaxe příkazu SQL je závislá na zdrojové databázi.

Element `<script>` obsahuje spustitelný kód v jazyce SQL nebo JavaScript. SQL skript je spuštěn nad připojením určeným v atributu `connection-id`. Pokud je element `script` vnořen v elementu `query` je možné v rámci napsaného kódu skriptu pracovat s daty výsledku výběrového dotazu, tato data transformovat a ukládat do cílové databáze. Syntaxe SQL skriptu je závislá na připojeném databázovém systému.

Element `<onerror>` umožní definovat skript, který bude spuštěn v případě, že ve spuštěném kódu dojde k výjimce.

Element `<include>` slouží k vložení obsahu dalšího definičního souboru na místo, ve kterém je element `include` použit. Element umožňuje složitější skript rozdělit do více souborů. [11]

2.3.2 Příklad importu dat pomocí aplikace Scriptella

Následující příklad demonstruje použití aplikace Scriptella při importu dat měřených stromů z tabulky databáze Access do tabulky databáze PostgreSQL. Ve dvou elementech `<connection>` jsou definovány připojovací řetězce pro zdrojovou databázi Access a cílovou databázi PostgreSQL. Zdrojový datový soubor Access je připojen prostřednictvím JDBC-ODBC mostu. Databáze PostgreSQL je připojena prostřednictvím rozhraní JDBC. V parametru `classpath` je uvedena cesta k souboru JDBC ovladače pro použitý databázový systém PostgreSQL verze 9.0.

V elementu `<script>` s připojením k cílové databázi systému PostgreSQL je vytvořena tabulka *tree* pro uložení měřených dat stromů. Pokud již tabulka *tree* existovala před spuštěním skriptu, bude skriptem smazána a vytvořena nová.

V elementu `<query>` s připojením ke zdrojové databázi Access jsou pomocí výběrového dotazu SQL vybrány záznamy měřených stromů určených pro import. Element `<query>` obsahuje vnořený element `<script>` s připojením k cílové databázi. Vnořený element obsahuje SQL příkaz pro vložení dat do tabulky v cílové databázi. SQL příkaz bude v cyklu spuštěn pro každý záznam získaný výběrovým dotazem v nadřazeném elementu `<query>`.

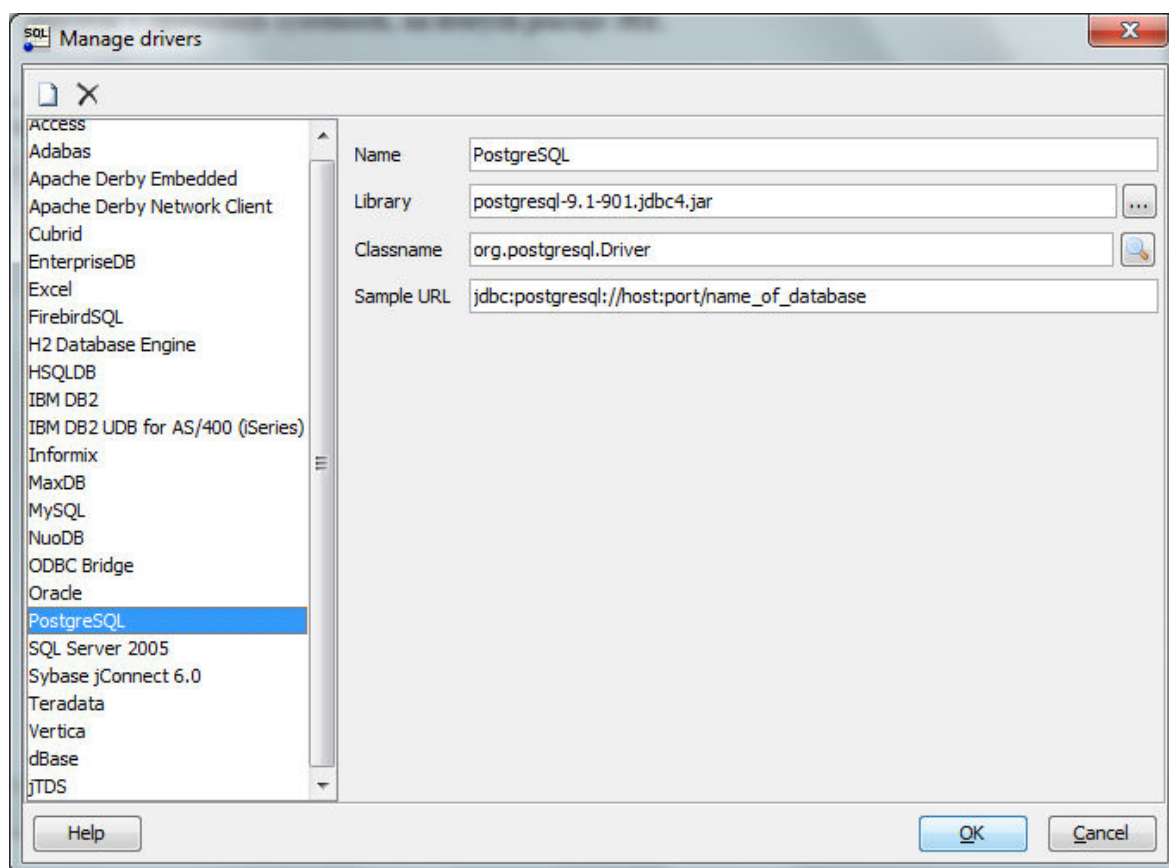
```
<?xml version="1.0"?>
<!DOCTYPE etl SYSTEM "http://scriptella.javaforge.com/dtd/etl.dtd">
<etl>
  <description> Import dat Access</description>
  <connection
    url="jdbc:odbc:DRIVER={Microsoft Access Driver
      (*.mdb)};DBQ=D:\source_project.mdb"
    id="odbc"/>
  <connection
    driver = "org.postgresql.Driver"
    classpath = "lib\postgresql-9.0.802.jdbc4.jar"
    url = "jdbc:postgresql://127.0.0.1:5432/DS_NIL"
    user = "postgres"
    password = "heslo"
    id = "pgsql" />
  <script connection-id = "pgsql">
    DROP TABLE IF EXISTS nfi_data.tree;
    CREATE TABLE nfi_data.tree (
      id_plot integer NOT NULL,
      id_tree integer NOT NULL,
      species integer NOT NULL,
      dbh integer,
      height double precision,
      age integer );
    ALTER TABLE nfi_data.tree
    ADD CONSTRAINT pk_tree PRIMARY KEY (id_plot, id_tree);
  </script>
  <query connection-id = "odbc">
    SELECT id_plot, id_tree, species, dbh, height, age
    FROM tree;
  <script connection-id = "pgsql">
    INSERT INTO nfi_data.tree
      (id_plot, id_tree, species, dbh, height, age)
    VALUES (?id_plot, ?id_tree, ?species, ?dbh, ?height,
      ?age);
  </script>
</query>
</etl>
```

2.4 SQL Workbench/J

SQL Workbench/J není ETL nástrojem v pravém slova smyslu. SQL Workbench/J je editor SQL dotazů. V rámci editovaných SQL dotazů však umožňuje spouštět své funkce pro import a export dat. SQL Workbench/J je programem, který byl vytvořen v programovacím jazyce Java. SQL Workbench/J je multiplatformní program, který může pracovat v operačních systémech, na kterých pracuje JRE.

2.4.1 Správa JDBC ovladačů v aplikaci SQL Workbench/J

SQL Workbench/J je aplikací, která je nezávislá na databázovém systému. Pro připojení a práci s databází používá JDBC ovladače pro zvolené databázové systémy. Prvním krokem při práci s aplikací je nastavení JDBC ovladačů pro databázové systémy mezi kterými data budou přenášena. Nastavení ovladače je provedeno zobrazením dialogu *Manage drivers* (Obr. 7) v nabídce *File*.



Obr. 7. SQL Workbench/J - dialog *Manage drivers*

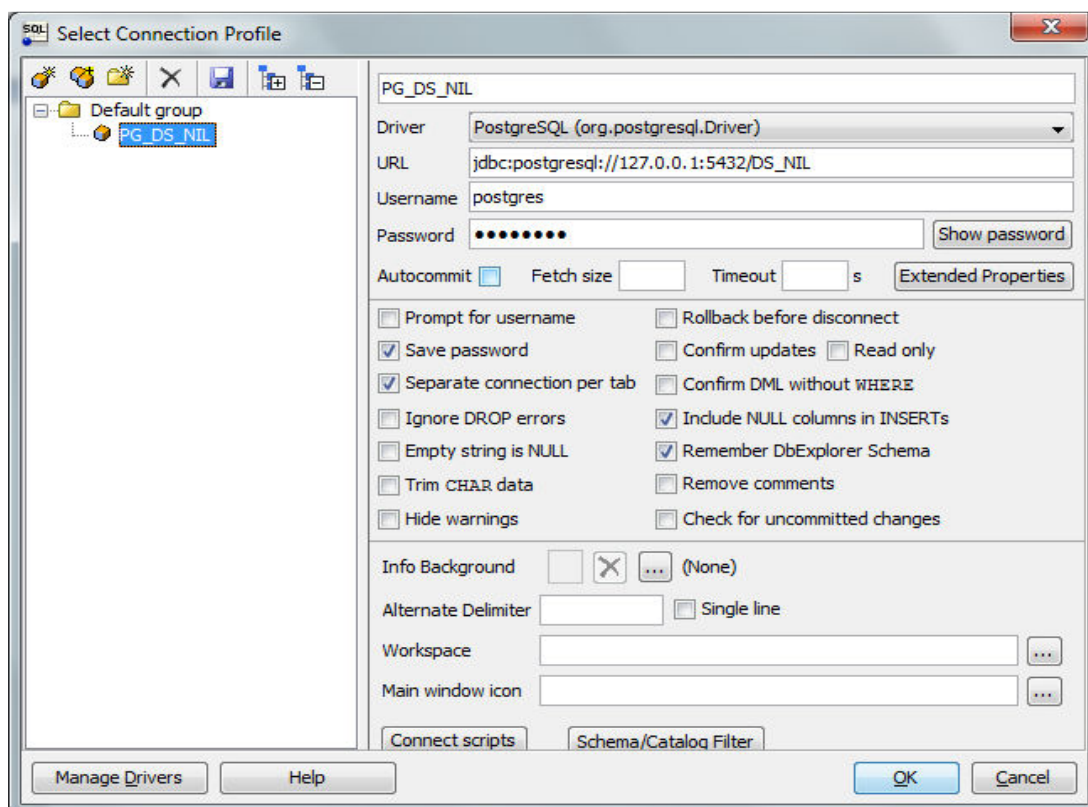
Při nastavení JDBC ovladače se určí jméno třídy ovladače a cesta k souboru jar, knihovně ve které se nachází třída ovladače.

Aplikace SQL Workbench/J umožňuje také připojení k databázovým systémům prostřednictvím rozhraní ODBC za použití JDBC-ODBC mostu. V tomto případě je nejdříve nutné v operačním systému nastavit a pojmenovat datový zdroj ODBC a tento datový zdroj použít k nastavení JDBC-ODBC mostu v aplikaci SQL Workbench/J.

2.4.2 Definice připojovacích profilů aplikace SQL Workbench/J

Pro práci s databázemi aplikace využívá připojovacích profilů. Připojovací profil uchovává informace pro připojení k databázovému serveru a informace specifikující chování a práci aplikace SQL Workbench/J s tímto připojením. Připojovacích profilů je možné definovat více k různým databázovým systémům. Vytvořený připojovací profil je pojmenován a uložen pro opakované použití.

Při spuštění aplikace je nejdříve zobrazen dialog *Select Connection Profile* (Obr. 8) se seznamem definovaných připojovacích profilů pro výběr aktivního připojení k databázi. Dialog umožňuje také definovat nové připojovací profily. Pro každý připojovací profil je nutné zvolit JDBC ovladač, určit databázový server IP adresou a TCP portem, a určit databázi a uživatele.



Obr. 8. SQL Workbench/J - dialog *Select Connection Profile*

2.4.3 Import dat aplikací SQL Workbench/J

Pro import dat v aplikaci SQL Workbench/J je určen příkaz *WbImport*. Příkaz umožňuje importovat data do zvolené tabulky databáze, která je určena aktivním připojovacím profilem. Pomocí příkazu je možné importovat data z textových souborů, ve kterých jsou sloupce určeny pevnou délkou nebo s definovaným znakem oddělovače. Příkazem je možné importovat data listů souboru tabulkového procesoru Microsoft Excel a data souborů tabulkového procesoru Calc aplikace OpenOffice. Příkaz umožňuje také načítat data ze souboru formátu xml, který vznikl exportováním dat v příkazu *WbExport*.

Chování příkazu *WbImport* je možné upravovat nastavováním mnoha parametrů příkazu. Příkaz umožňuje data importovat v módu insert nebo v módu update. V módu insert jsou pro načtené datové řádky zdrojového souboru generovány insert příkazy pro cílovou datovou tabulku. Insert mód tak umožňuje vkládání nových řádků z textového souboru do databázové tabulky. V módu update aplikace SQL Workbench/J generuje pro načítané řádky zdrojového souboru update příkazy pro cílovou databázovou tabulku. V update módu je tak možné již existující datové záznamy v databázové tabulce aktualizovat podle obsahu importovaného souboru. Aby bylo možné jednoznačně identifikovat datové záznamy v cílové tabulce, update mód vyžaduje určení jmen datových sloupců, které tvoří primární klíč nebo sloupců, které obsahují pro každý záznam unikátní hodnotu a umožňují tak záznam pro update jednoznačně identifikovat. Tyto zvolené sloupce musí rovněž obsahovat zdrojový datový soubor. V případě, že zdrojový soubor obsahuje nové záznamy zároveň se záznamy, které se již v tabulce vyskytují je možné záznamy importovat v kombinovaném módu, který zajistí vložení nových záznamů a aktualizaci existujících. [12]

Příkazem *WbImport* jsou načítána data jednoho datového souboru nebo více datových souborů, které jsou uloženy ve zdrojovém adresáři. Při načítání více souborů příkaz vyžaduje, aby jména souborů se shodovala se jmény cílových tabulek. [12]

Pomocí nastavení parametrů příkazu *WbImport* je možné importovaná data filtrovat, volit pouze některé sloupce pro import, nastavovat defaultní hodnoty v datových sloupcích, provádět jednoduché transformace, nahrazovat prázdné řetězce NULL hodnotou, ořezávat prázdné znaky na koncích textových řetězců, zkracovat textové řetězce na definovanou délku, nahrazovat logické hodnoty textem nebo číslem, nahrazovat části textových řetězců. [12]

Pokud cílová tabulka obsahuje definovaná check omezení, která importovaná data kontrolují, pak je možné v parametru *-badFile* určit vytvoření souboru pro uložení chybných záznamů, které porušují dané omezení a příkazu *WbImport* se nepodařilo tento záznam do databázové tabulky importovat.

V následujícím příkladu jsou importována měřená data stromů z textového souboru určeného v parametru *-file*. Textový soubor obsahuje v prvním řádku popisky sloupců a tento řádek bude při importu dat přeskočen (parametr *header* = true). Data v textovém souboru jsou oddělena znakem středníku (parametr *delimiter*). Parametr *-importColumns* určuje jména sloupců, které budou z textového souboru importovány. Záznamy budou uloženy do tabulky *tree* ve schématu *nfi_data*. V případě, že záznam je nový, bude do tabulky vložen. V opačném případě se provede aktualizace existujícího záznamu (parametr *mode*). Záznamy budou jednoznačně určeny identifikačním číslem plochy (*id_plot*) a identifikačním číslem stromu (*id_tree*), což je definováno v parametru *keyColumns*.

WbImport

```
-type = text
-mode = 'INSERT,UPDATE'
-file = D:/dp/tree_data.csv
-header = true
-delimiter = ';'
-schema = nfi_data
-table = tree
-keyColumns = id_plot, id_tree
-importColumns = id_plot, id_tree, species, dbh, height, age
```

2.4.4 Příklad exportu dat aplikací SQL Workbench/J

Pro export dat je v aplikaci SQL Workbench/J určen příkaz *WbExport*. Příkazem je možné exportovat data z tabulek databázového systému určeného aktivním připojovacím profilem do mnoha různých formátů. Příkaz podporuje export do textových souborů s definovanými oddělovači datových sloupců. Exportovat data je možné do souborů xml nebo také, při doplnění softwarových knihoven, do formátu aplikace Microsoft Excel. Textový soubor může být doplněn o první řádek s hlavičkou se jmény datových sloupců. Ve vytvořených textových souborech data nemusí být zapsána pouze ve formě řádků s definovanými konci a oddělovači hodnot. Příkaz *WbExport* umožní vytvořit pro jednotlivé exportované záznamy tabulek SQL příkazy INSERT, UPDATE a DELETE.

Tyto je možné zapsat do výsledného textového souboru a import dat později provést spuštěním tohoto souboru. Protože provedení velkého množství INSERT příkazů má vliv na výkon databázového systému, mají databázové systémy nástroje pro import dávky dat, provedení BULK INSERT. V případě databázového systému PostgreSQL se jedná o nástroj COPY. Existuje proto tedy možnost data exportovat ve formátu vhodném pro import daným nástrojem zvoleného cílového databázového systému. Výsledný soubor může být také komprimován do zip archivu. Výběr dat pro export je určen jménem schématu a tabulky, jejíž data mají být exportována. Výběr dat je možné omezit zadáním výběrové podmínky v parametru *tableWhere*. V případě, že není nutné exportovat data celé tabulky, pak zadáním výběrového dotazu SELECT, který následuje za příkazem *WbExport* můžeme do souboru exportovaných dat zapsat pouze výsledek výběrového dotazu. Exportovaná data je možné zadáním parametrů příkazu *WbExport* modifikovat, upravovat formáty časových údajů, specifikovat desetinné oddělovače čísel, aplikovat regulární výrazy a nahrazovat části textových řetězců. [12]

Následující příklad provádí export dat z databázové tabulky *tree* (parametr *sourcetable*), schématu *nfi_data*. Exportovaná data omezuje na záznamy buku lesního (*species* = 50) o výčetní tloušťce kmene větší 180 mm. Výsledek zapíše do textového souboru (parametr *type*) *tree_export.zip*. Do textového souboru bude vložen první řádek s názvy sloupců (parametr *header*), data budou oddělena znakem středníku (parametr *delimiter*), desetinná čísla budou mít desetinnou část oddělenou tečkou (parametr *decimal*). Textový soubor bude uložen v kódování UTF8. Celý soubor bude komprimován do zip archivu (parametr *compress*).

WbExport

```
-type = text
-file = D:/dp/tree_export.txt
-sourcetable = nfi_data.tree
-tableWhere = 'WHERE species = 50 AND dbh >= 180'
-header = true
-delimiter = ';'
-decimal = '.'
-lineEnding = crlf
-encoding = UTF8
-compress = true;
```


2.4.5 Migrace dat datové tabulky aplikací SQL Workbench/J

Pro přenos dat z datové tabulky databáze zdrojového systému do datové tabulky cílového systému je určen příkaz *WbCopy*. Pro připojení k databázím je nejdříve nutné vytvořit dva pojmenované připojovací profily pro zdrojový a cílový systém a připojovací profil zdroje a cíle předat příkazu *WbCopy* v parametrech *sourceProfile* a *targetProfile*.

Podobně jako tomu bylo v příkazu *WbImport*, příkaz *WbCopy* umožňuje nahrání dat provést v módu insert, update nebo v kombinovaném módu. V případě update módu je nezbytně nutné určit v parametru *keyColumns* sloupce primárního klíče, které záznamy v tabulkách jednoznačně identifikují. Nastavením kombinovaného módu lze dosáhnout synchronizace dat mezi tabulkami zdrojového a cílového systému. Provedení příkazu vloží nové záznamy do tabulky cílového systému a aktualizuje záznamy existující na nové hodnoty obsažené ve zdroji. Problém nastává, pokud ve zdrojovém systému některé datové záznamy byly vymazány. Při nastavení parametru *syncDelete* příkazu *WbCopy* aplikace prohledá cílovou tabulku a podle hodnoty primárního klíče záznamy porovná s tabulkou zdroje. Pokud cílová tabulka obsahuje záznamy, které nejsou obsaženy ve zdrojové tabulce, budou nalezené záznamy vymazány. V případě, že názvy sloupců zdrojové tabulky a cílové tabulky jsou rozdílné, je nutné definovat mapování sloupců. Mapování sloupců je definováno v parametru *columns*, ve kterém je uveden seznam párů sloupců, jejichž data budou kopírována. Každý pár určuje zdrojový a cílový sloupec oddělené znakem lomítka. [12]

V některých případech zdrojem dat pro cílovou tabulku nemusí být pouze jedna tabulka zdrojového systému. Příkaz *WbCopy* umožní v parametru *sourceQuery* definovat výběrový dotaz v jazyce SQL. Tento výběrový dotaz je spouštěn na zdrojovém databázovém systému. Záznamy pro import do cílové tabulky jsou pak určeny výsledkem tohoto výběrového dotazu. [12]

V následujícím příkladu byly vytvořeny dva připojovací profily PG_DS_NIL určující zdrojovou databázi v databázovém systému PostgreSQL a PG_DS_NIL_analytical určující pracovní cílovou databázi v databázovém systému PostgreSQL. Příkaz *WbCopy* spustí výběrový SQL dotaz specifikovaný v parametru *sourceQuery*. Dotaz provede agregaci záznamů z tabulek *nfi_data.tree* a *nfi_data.plot* a určí počet záznamů a průměrnou hodnotu výšky stromů v agregovaných skupinách určených dimenzemi druh dřeviny a kraj. Výsledek uloží do tabulky *tree_summary* v druhé databázi určené cílovým připojovacím

profilem. Protože cílová tabulka již záznamy obsahuje, bude obsah tabulky aktualizován novými hodnotami nastavením kombinovaného módu přenosu dat. Protože očekáváme větší počet příkazů UPDATE než INSERT, pak z důvodu rychlejšího provedení příkazu *WbCopy* předřadíme v kombinovaném módu příkaz UPDATE před příkaz INSERT.

WbCopy

```
-sourceProfile = PG_DS_NIL
-targetProfile = PG_DS_NIL_analytical
-mode = 'UPDATE,INSERT'
-sourceQuery =
'SELECT t.species, p.district,
        count(t.id_tree) AS tree_count,
        round(avg(t.height),2) AS mean_height
FROM nfi_data.tree AS t
LEFT JOIN nfi_data.plot AS p
        ON (p.id_plot = t.id_plot)
WHERE t.height IS NOT NULL
GROUP BY t.species, p.district;'
-targetTable = nfi_data.tree_summary
-keyColumns = species, district
-columns = species, district, tree_count, mean_height;
```

3 NÁSTROJE PRO IMPORT A EXPORT DAT DATABÁZOVÉHO SYSTÉMU POSTGRESQL

Databázový systém PostgreSQL obsahuje příkazy a rozšíření pro provádění importů a exportů dat. V mnoha případech, ve kterých není potřeba řešit složité transformace importovaných dat, může být použití externího nástroje zbytečné a složité, protože import dat je možné rychle provést přímo pomocí nástroje databázového systému.

3.1 Příkaz COPY

Příkaz COPY v databázovém systému PostgreSQL slouží k importu dat uložených v textových souborech do vybrané tabulky databáze a k exportu dat tabulky do textových souborů. Varianta příkazu COPY FROM importuje data z textového souboru do databázové tabulky, varianta COPY TO exportuje data z vybrané tabulky do textového souboru. Příkaz COPY je možné spouštět z *psql*, které je příkazovým řádkem databázového systému PostgreSQL.

Příkaz COPY také může být použit jako možnost pro sdílení dat s dalšími aplikacemi nebo jako rychlá záloha dat vybrané tabulky. [13]

V následujícím příkladu je ukázán export dat tabulky stromů *tree* z databáze inventarizace lesů do textového souboru a import dat stromů z textového souboru do tabulky *tree*:

```
COPY nfi_data.Tree TO 'D:/tree.txt'
WITH DELIMITER ';'
COPY nfi_data.Tree FROM 'D:/tree.txt'
WITH DELIMITER ';'

```

V defaultním nastavení jsou v exportovaném textovém souboru hodnoty odděleny znakem horizontálního tabulátoru a prázdné hodnoty NULL jsou nahrazeny znakem \N. Defaultní chování příkazu je možné změnit nastavením voleb za příkazem WITH.

Příkazem COPY je možné exportovat nejen data celé tabulky, ale také určit sloupce nebo exportovat výsledek výběrového dotazu:

```
COPY (SELECT id_plot, id_tree, dbh, height, age
      FROM nfi_data.Tree
      WHERE species = 1)
TO 'D:/picea_abies.txt'
WITH DELIMITER ';'

```

Příkaz COPY pro import a export dat používá soubory v lokálním souborovém systému a není možné ho použít k importu a exportu dat na vzdáleném serveru při přístupu prostřednictvím počítačové sítě. [13]

Část příkazu obsahující jméno souboru může být nahrazena příkazy *stdin* a *stdout*, které představují vstup a výstup programu *psql*. Použití příkazů *stdin* a *stdout* umožní vypisovat výsledek příkazu COPY na obrazovku nebo zadávat vstupní data z klávesnice. [13]

3.2 Rozšíření DBLINK

Dblink je modul obsahující rozšiřující funkce databázového systému PostgreSQL. Funkce modulu *dblink* umožňují vytvářet připojení ke vzdáleným databázovým serverům PostgreSQL. Na vytvořeném vzdáleném připojení je možné potom vytvářet a spouštět výběrové dotazy. Rozšíření *dblink* tak můžeme použít k importu dat z jiného databázového serveru PostgreSQL a to tak, že výsledky výběrového dotazu budou jednoduše vloženy do tabulek databáze na lokálním serveru. Druhou možností využití funkcí tohoto rozšíření je použití v dotazech na data jiné databáze na tomtéž databázovém serveru. Na rozdíl od jiných databázových systémů, databázový systém PostgreSQL nedovoluje vytváření dotazů mezi databázemi, ale pouze v rámci téže databáze. Toto omezení je možné obejít právě použitím funkcí modulu *dblink*.

Funkce modulu *dblink* jsou uloženy v souboru *dblink.sql* v adresáři *share/extension* a před jejich použitím je nutné je nainstalovat a zavést pomocí příkazu CREATE EXTENSION *dblink*.

V následujícím příkladu je ukázáno vytvoření připojení k vzdálenému databázovému serveru, který je identifikovaný svojí IP adresou. Z databáze DS_NIL na vzdáleném serveru budou vybrána všechna data tabulky stromů *tree* ve schématu *nfi_data* a tato data budou uložena v kopii na lokálním serveru:

```
PERFORM dblink_connect
(
    'connection',
    'host = 192.168.192.25
    user = postgres
    password = heslo
    dbname = DS_NIL'
);
INSERT INTO nfi_data.tree
```

```
(
    id_plot, id_tree, dbh, height, species, age
)
SELECT
    id_plot, id_tree, dbh, height, species, age
FROM dblink
(
    'connection',
    'SELECT
    id_plot, id_tree, dbh, height, species, age
    FROM nfi_data.tree'
)
AS result_table
(
    id_plot            integer,
    id_tree            integer,
    dbh                integer,
    height             double precision,
    species            integer,
    age                integer
);
PERFORM dblink_disconnect ('connection');
```

Prvním krokem v příkladu bylo vytvoření připojení ke vzdálenému databázovému serveru pomocí příkazu *dblink_connect*, ve kterém je nové připojení pojmenováno jménem *connection* a je určen databázový server a databáze, ke které je připojení vytvořeno. Připojení je vytvářeno pro určitého uživatele, který se ověřuje svým heslem.

Druhým krokem je pak vložení výsledku výběrového dotazu do místní datové tabulky *tree*. Prázdnou místní datovou tabulku *tree* je nutné si vytvořit před spuštěním přenosu dat. Výběrový dotaz nemá v části FROM určenou místní datovou tabulku, která je nahrazena příkazem *dblink*. Příkaz *dblink* má dva parametry, kterými jsou jméno připojení vytvořené v prvním kroku a výběrový dotaz, který je proveden na zdroji dat v databázi na vzdáleném serveru. Výsledkem příkazu *dblink* je databázové tabulka, kterou je nutné definovat za příkazem AS.

Drobnou nevýhodou použití *dblink* je nutnost v každém příkazu znovu definovat strukturu výsledné datové tabulky. V případě příkazu, který je často používán je proto výhodné uložit tento dotaz do databáze ve formě datového pohledu na data uložená na jiném

databázovém serveru nebo jako uloženou proceduru, která tato data bude vracet ve formě datové tabulky.

Posledním krokem je uzavření vytvořeného připojení pomocí příkazu *dblink_disconnect*.

3.3 Import a export binárních dat

Při šetření na inventarizačních plochách jsou kromě numerických dat pořizovány také různé druhy dokumentačních fotografií. Při návrhu technologie skladování dat inventarizace lesů bude třeba řešit problém jak uchovávat tyto fotografie s vazbami na příslušná numerická a textová data.

Databázový systém PostgreSQL nabízí tři možnosti přístupu k binárním datům, z nichž každá má své výhody i nevýhody.

První možností je uložení binárních dat v souborovém systému. Databáze pak obsahuje u jednotlivých záznamů pouze uložení cesty k souboru. Vlastní binární data jsou uložena mimo databázi. Nevýhodou je, že takto uložená binární data nejsou pod správou databázového systému. Databázový systém nemůže řídit přístup uživatelů k takto uloženým souborům. Pokud dojde ke smazání nebo změně jména souboru, v databázové tabulce zůstává neplatný odkaz na neexistující datový soubor.

Druhou možností je definovat v rámci datové tabulky sloupec datového typu *varbinary*. Soubory fotografií je pak možné ukládat přímo v záznamu v databázové tabulce s příslušnými numerickými daty. Binární data uložená v datové tabulce rychle zvětšují velikost tabulky.

Třetím kompromisním řešením je uchování binárních dat jako *large object*. Binární data *large objektů* jsou ukládána v systémové tabulce katalogu (*pg_catalog*), která se jmenuje *pg_large_object*. Při uložení objektu do tabulky katalogu je objekt rozdělen do několika částí definované velikosti a každá část je uložena jako samostatný datový řádek v tabulce *pg_large_object*. Každý *large object* má své jednoznačné identifikační číslo, které se označuje *oid*. [14] V datových tabulkách jsou pak u příslušných záznamů uchovávány hodnoty *oid* představující reference na příslušné *large objekty* v systémovém katalogu.

Pro každý *large objekt* podobně jako u jiných databázových objektů je možné definovat jeho vlastníka a nastavit oprávnění pro jednotlivé uživatele a skupiny. Oprávnění jsou ukládána do systémové tabulky katalogu *pg_large_object_metadata*. [14]

Databázový systém poskytuje řadu funkcí pro práci s large objekty. Jsou jimi funkce pro otevření a uzavření large objektu, funkce pro jeho vytvoření a zrušení, funkce pro čtení a zápis dat do large objektu.

Pro import a export dat binárních souborů ze souborového systému do databáze je určena dvojice funkcí *lo_import* a *lo_export*. Následující příklad je ukázkou importu souboru fotografie půdní sondy a její registrace v tabulce *sond_photo*:

```
INSERT INTO nfi_data.sond_photo
(id_plot, sond_photo_data)
VALUES
(196967, lo_import ('D:/IMG_196967.jpg'));

ALTER LARGE OBJECT 1007903
OWNER TO adm_nfi_data;

GRANT SELECT ON LARGE OBJECT 1007903
TO usr_nfi_data;

SELECT lo_export (1007903, 'D:/IMG_196967_exp.jpg');
```

V první části příkladu byl do tabulky fotografií půdních sond *sond_photo* vložen nový záznam fotografie půdní sondy na inventarizační ploše určený jejím *id_plot*. V rámci příkazu *insert* byla volána funkce *lo_import*, která importuje data souboru *IMG_196967.jpg* jako large object do systémové tabulky katalogu a vrací *oid* nově vytvořeného large objektu, který je zapsán do tabulky *sond_photo*.

Vlastníkem nově vytvořeného large objektu je uživatel, který jej vytvořil. Následuje proto příkaz, který nastavuje nového vlastníka large objektu identifikovaného pomocí jeho *oid* a příkaz pro nastavení oprávnění čtení large objektu pro skupinu uživatelů.

Posledním příkazem příkladu je export large objektu z databáze do souborového systému. Export je proveden pomocí funkce *lo_export*, která přijímá dva parametry, kterými jsou identifikátor large objektu a cesta, kde bude nový soubor fotografie uložen.

4 PŘÍSTUPY K DATABÁZOVÝM SYSTÉMŮM

Úkolem aplikace pro migraci dat inventarizace lesů je načítat změřená data z datových souborů ve formátu databáze Access a tato načtená data ukládat na jednotné datové úložiště postavené na databázovém systému PostgreSQL. Při tvorbě aplikace je tedy nutné řešit volbu metody přístupu k datům v souborech Access a metody přístupu k databázi v systému PostgreSQL. Použitý programovací jazyk C# nabízí několik možností přístupu k datům v databázích PostgreSQL.

4.1 ODBC

ODBC představuje standardizované softwarové rozhraní mezi databázovým systémem a aplikací. Jeho cílem je, aby přístup k databázovému systému byl jednotný a nezávislý na použitém databázovém systému a programovacím jazyce, ve kterém je aplikace vytvořena. Architektura ODBC je třívrstvá. Aplikace komunikuje s databází prostřednictvím funkcí knihovny ODBC. Komunikace s datovým zdrojem je zajištěna ovladačem, který je specifický pro daný datový zdroj. [15]

Pro použití ODBC je nutné nejdříve ověřit, zda na počítači, na kterém bude spouštěna aplikace vyžadující přístup k databázi, je nainstalován ODBC ovladač pro databázový systém PostgreSQL. V případě prvního použití je třeba tento ovladač nainstalovat. Pokud se jedná o počítač, na kterém je spuštěna instance databázového serveru PostgreSQL, může být ovladač nainstalován již při instalaci databázového serveru. Instalační program databázového serveru nabízí volbu instalace ODBC ovladače.

Při tvorbě aplikace v programovacím jazyce C# v platformě .NET jsou pak pro práci s datovými zdroji ODBC dostupné a využitelné standardní objekty z tříd, které se jsou definovány ve jmenném prostoru *Microsoft.Data.Odbc*.

Prvním krokem při práci s databází systému PostgreSQL v C# je vytvoření připojení k datovému zdroji. Pro vytvoření připojení slouží objekt třídy *OdbcConnection*. Pro specifikaci serveru a databáze, ke které se aplikace připojuje, je nutné nastavit objektu *OdbcConnection* vlastnost *ConnectionString* nebo tuto vlastnost určit při vytváření objektu v jeho konstruktoru. [16]

Druhou možností je definovat připojení ODBC prostřednictvím správce zdrojů dat ODBC operačního systému Windows. V konstruktoru objektu třídy je pak předáno pouze jméno předdefinovaného datového zdroje DSN. [16]

Pro připojení k databázovým systémům PostgreSQL má připojovací řetězec následující tvar:

```
OdbcConnection connection = new OdbcConnection();  
connection.ConnectionString = String.Concat(  
    "DRIVER {PostgreSQL}; SERVER = 127.0.0.1; ",  
    "UID = postgres; PSW = heslo; DATABASE = DS_NIL");
```

V připojovacím řetězci definujeme použitý ODBC ovladač, jméno nebo IP adresu databázového serveru, číslo TCP portu, jméno a heslo přihlašovaného uživatele a jméno databáze, ke které se aplikace připojuje.

Rozhraní ODBC je možné použít také k připojení zdrojových souborů databáze Access:

```
OdbcConnection connection = new OdbcConnection();  
connection.ConnectionString = String.Concat(  
    "DRIVER {Microsoft Access Driver (*.mdb)}; ",  
    "DBQ = D:\\dp\\source.mdb; ");
```

Při práci s databází je nutné nejdříve definované připojení k databázi otevřít použitím metody *Open* objektu třídy *OdbcConnection*. Soubor tříd knihovny *Microsoft.Data.Odbc* pak obsahuje definice řady tříd využívajících vytvořené připojení a umožňující práci s daty a objekty databáze. Po dokončení práce je vytvořené připojení uzavřeno pomocí metody *Close* objektu třídy *OdbcConnection*.

Aplikace pro migraci dat používá rozhraní ODBC pro přístup k zdrojovým datovým souborům ve formátu databáze Access. Pro spouštění SQL příkazů jsou používány objekty třídy *OdbcCommand*. Objekty třídy *OdbcCommand* umožňují spuštění příkazu uloženého ve vlastnosti *CommandText* nad připojením určeným objektem třídy *OdbcConnection* specifikovaného ve vlastnosti *Connection*. Objekty třídy *OdbcCommand* obsahují metody pro spuštění definovaného SQL příkazu. Používány jsou metody *ExecuteNonQuery* v případě, že není vrácen žádný výsledek SQL příkazu. V případě, že SQL příkaz vrací výsledek výběrového dotazu je použita metoda *ExecuteReader*, která vrací objekt třídy *OdbcDataReader*. Objekt třídy *OdbcDataReader* je určený ke čtení záznamů získaných načtením z databáze. Metoda *Read* objektu třídy *OdbcDataReader* pak umožňuje sekvenční čtení vrácených záznamů výsledku výběrového dotazu. Pro plnění objektů datových tabulek je používána metoda *Fill* objektu třídy *OdbcDataAdapter*.

4.2 Npgsql

Npgsql je knihovna tříd pro připojení a poskytování dat databázového systému PostgreSQL pro klientské aplikace .NET.

Pro použití ve vývojovém prostředí Visual Studio je nejdříve nutné přidat referenci na dynamicky linkovanou knihovnu *Npgsql.dll*. Definice tříd se nacházejí ve jmenném prostoru *Npgsql*.

K připojení k databázi PostgreSQL je určen objekt třídy *NpgsqlConnection* v jehož konstruktoru nebo nastavením vlastnosti *ConnectionString* je specifikován připojovací řetězec. Připojovací řetězec obsahuje určení databázového serveru PostgreSQL jeho IP adresou a TCP portem, dále je uvedeno jméno přihlašovací role, heslo a jméno databáze. Připojení k databázi je provedeno spuštěním metody *Open* objektu třídy *NpgsqlConnection*. Po ukončení práce s databází je nutné otevřené připojení uzavřít voláním metody *Close*.

```
NpgsqlConnection connection = new NpgsqlConnection();  
connection.ConnectionString = String.Concat(  
    "Server = 127.0.0.1; Port = 5432; User Id = postgres; ",  
    "Password = heslo; Database = DS_NIL");
```

Třídy poskytovatele dat *Npgsql* implementují rozhraní ze jmenného prostoru *System.Data*. To znamená, že při práci s knihovnou *Npgsql* jsou k dispozici všechny třídy a jejich metody definované v rozhraní knihovny ADO.NET, podobně jak tomu bylo u poskytovatele dat ODBC. Knihovna *Npgsql* navíc definuje třídy pro práci s *large objects* a třídy pro import a export dat prostřednictvím příkazu COPY, které jsou specifické pro databázový systém PostgreSQL.

K práci s binárními daty large objektů poskytovatel dat *Npgsql* nabízí definice tříd *LargeObjectManager* a *LargeObject*. Při zápisu binárních dat do databáze PostgreSQL je postupováno v několika krocích. Nejdříve je vytvořen nový objekt třídy *LargeObjectManager* a v jeho konstruktoru je určeno připojení k databázi PostgreSQL prostřednictvím objektu třídy *NpgsqlConnection*. Poté je volána metoda *Create* objektu třídy *LargeObjectManager*, která v databázi vytvoří nový large objekt a vrátí jeho identifikační číslo:

```
LargeObjectManager manager = new LargeObjectManager(connection)
```

```
int oid = manager.Create(LargeObjectManager.READWRITE)
```

Pomocí metody *Open* třídy *LargeObjectManager* je nový objekt třídy *LargeObject* otevřen pro čtení a zápis:

```
LargeObject lObj = manager.Open(oid, LargeObjectManager.READWRITE);
```

Objekt třídy *LargeObject* umožňuje zápis binárních dat ve formě pole bytů pomocí metody *Write*. Po dokončení zápisu je objekt uzavřen metodou *Close*:

```
lObj.Write(byte_array);
```

```
lObj.Close();
```

Pro čtení binárních dat z databáze je určena metoda *Read* objektu třídy *LargeObject*, která načtená data vrátí ve formě pole bytů:

```
byte[] byte_array);
```

```
byte_array = lObj.Read(lo.Size());
```

Objekt třídy *LargeObjectManager* umožňuje odstranění existujících large objektů z databáze pomocí metody *Delete* a známého identifikačního čísla large objektu:

```
manager.Delete(oid);
```

II. PRAKTICKÁ ČÁST

5 NÁVRH DATABÁZE

Následující část se zabývá popisem tabulek databáze, které jsou určeny pro uložení měřených dat. Ve skutečnosti měření v druhém cyklu inventarizace probíhá na dvou inventarizačních sítích a to na původní síti prvního cyklu a na nově navržené inventarizační síti pro druhý cyklus měření. Rozsah měření a seznam měřených veličin je pro obě sítě různý a pro uložení měřených dat byly vytvořeny dvě samostatné databáze pro každou z inventarizačních sítí. Aplikace pro migraci dat inventarizace lesů musí umožnit nahrání měřených dat obou inventarizačních sítí. Cílem této části práce není tedy připravení databází pro obě sítě, ale spíše příprava testovací databáze, která bude určena pro vývoj a testování aplikace pro migraci dat.

Při návrhu databáze se vychází z navržených metodik měření, které určují, jaké objekty v lesních porostech budou popisovány a jaké veličiny tyto objekty popisují. Popisované objekty jsou potom v návrhu databáze představovány tabulkami (relacemi), měřené veličiny jsou datovými sloupci (atributy). Mezi popisovanými objekty existují vzájemné vazby. Například stromy rostou na inventarizační ploše, některé stromy jsou vzorníky, kmenový profil se zjišťuje na vzorníku a mnoho dalších. Tyto vazby jsou pak modelovány relačními vazbami mezi tabulkami tvořící databázi měření inventarizace lesů.

Při návrhu tabulky je také nutné rozhodnout o použití primárního klíče. První možností je použití složeného primárního klíče, ve kterém budou použity existující datové sloupce. Při sběru dat jsou měřeným objektům v rámci inventarizační plochy přiřazena identifikační čísla, která je možné použít ke konstrukci složeného primárního klíče. Druhou možností je použití jednoduchého primárního klíče, doplněním sloupce id, ke kterému je vázán objekt sekvence. Sekvence potom generuje hodnotu identifikátoru při každém vložení dalšího záznamu zvýšením o definovaný přírůstek. Výhodou jednoduchého klíče je rychlejší vyhledávání záznamu v tabulce podle hodnoty klíče, protože index, který je vytvářen pro každý primární klíč, obsahuje pouze jeden atribut. Na druhou stranu složený primární klíč nese určitou informaci o hierarchii tabulek a usnadňuje tak psaní výběrových dotazů, protože pro dohledávání záznamů v nadřazených tabulkách není nutné procházet celou hierarchii tabulek přes všechny úrovně zanoření.

Při měření na inventarizačních plochách jsou data zjišťována jednou měřicí skupinou ukládána do projektu měření. Měřicí skupiny projekty zasílají v týdenním intervalu. Měřená data jsou pak uložena ve velkém množství souborů. V rámci datového skladu

databáze měření na sítích prvního a druhého cyklu tvoří primární datové úložiště. Data v těchto databázích jsou uložena v normalizované podobě s minimální redundancí dat. Účelem těchto databází je centralizovat všechna měřená data do jednoho datového úložiště, které je zdrojem dat pro databáze, které jsou určeny pro konkrétní analýzy.

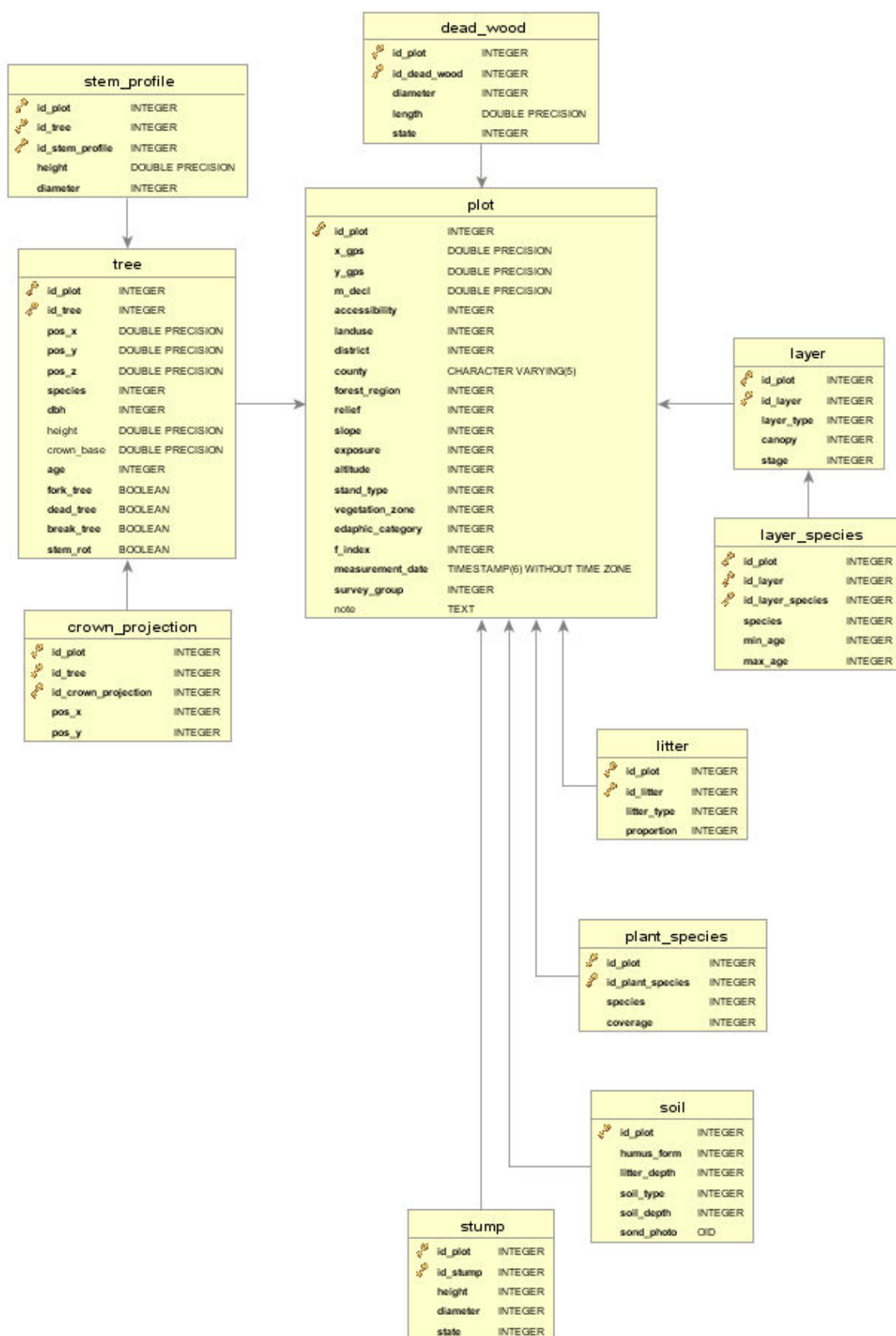
5.1 Seznam datových tabulek

Datové tabulky testovací databáze, jejich atributy a relační vazby mezi datovými tabulkami jsou popsány v diagramu databáze (Obr. 9).

- *plot* je tabulka obsahující seznam inventarizačních ploch, na kterých probíhá měření. Atribut *id_plot* představuje identifikační číslo plochy. Střed každé plochy je určen zaměřenými souřadnicemi (*x_gps*, *y_gps*), v souřadném systému S-JTSK. Atribut *m_decl* uchovává magnetickou deklinaci. Atribut *accessibility* určuje, zda bod je přístupný pro měření, *landuse* určuje, zda je bod v lese. Umístění bodu do kraje a okresu popsují atributy *district* a *county*. Zařazení bodu do přírodní lesní oblasti určuje atribut *forest_region*. Každá plocha je dále popsána kategorií reliéfu terénu (*relief*), sklonem svahu (*slope*), expozicí (*exposure*), nadmořskou výškou (*altitude*). Pro plochu je určen lesní typ složený z atributů vegetační stupeň (*zone*), edafická kategorie (*edaphic_category*) a fytocenologický index (*f_index*). Pro každou plochu je uvedeno číslo skupiny, která měření provedla (*survey_group*) a datum měření (*measurement_date*). Plochy je možné doplnit poznámkou z měření (*note*).
- *tree* je tabulka obsahující záznamy měřených stromů. Stromy rostou na inventarizačních plochách. Tabulka je proto spojena relační vazbou 1:N s tabulkou ploch. Každý záznam stromu je jednoznačně identifikován dvojicí identifikační číslo plochy a identifikační číslo stromu (*id_plot* a *id_tree*). Poloha stromu je určena souřadnicemi *pos_x*, *pos_y* a *pos_z* vztaženými ke středu plochy. Pro každý strom je určen jeho botanický druh (*species*). Změřena je výška stromu (*height*), výška koruny (*crown_base*), výčetní tloušťka (*dbh*) a věk stromu (*age*). Logické hodnoty určují souš (*dead_tree*), rozdvojený strom (*fork_tree*), zlomený strom (*break_tree*) a strom napadený hnilobou kmene (*stem_rot*).
- *stem_profile* je tabulka obsahující měření tloušťky kmene (*diameter*) v určených výškách na kmeni stromu (*height*). Měření je prováděno pouze na vybraných

stromech (vzornících), tabulka je tedy relační vazbou 1:N spojena s tabulkou stromů.

- *crown_projection* je tabulka korunových projekcí. Tabulka obsahuje souřadnice bodů korunové projekce stromu (*pos_x*, *pos_y*) vztažené ke středu plochy. Měření se týká vzorníků, tabulka je relační vazbou 1:N spojena s tabulkou stromů.
- *layer* je tabulka obsahující popis porostních vrstev. Tabulka obsahuje atributy typ porostní vrstvy (*layer_type*), procento zápoje (*canopy*) a stádium porostní vrstvy (*stage*). Porostní vrstvy jsou popisovány v rámci plochy, tabulka je tedy připojena relační vazbou 1:N k tabulce ploch.
- *layer_species* je tabulka obsahující seznam druhů dřevin tvořících porostní vrstvu. Pro každý druh dřeviny je uveden jeho botanický druh (*species*) a minimální a maximální věk (*min_age* a *max_age*). Každá porostní vrstva může obsahovat více druhů dřevin, tabulka je proto spojena s tabulkou *layer* relační vazbou 1:N.
- *plant_species* je tabulka obsahující seznam botanických druhů rostlin zjištěných na měřené ploše. Pro každý druh je uveden jeho botanický název (*species*) a četnost výskytu (*coverage*). Tabulka je spojena relační vazbou 1:N s tabulkou ploch.
- *soil* je tabulka obsahující popis půdní sondy. Tabulka zahrnuje atributy forma nadložního humusu (*humus_form*) a jeho tloušťka (*litter_depth*), půdní typ (*soil_type*), hloubku půdy (*soil_depth*) a typ matečné horniny (*parent_material*).
- *stump* je tabulka, která obsahuje měření nalezených pařezů. Každý pařez je popsán výškou (*height*), tloušťkou (*diameter*) a stupněm rozkladu mrtvého dřeva (*state*).
- *dead_wood* je tabulka záznamů nalezeného mrtvého dřeva, přesahující průměrem registrační hranici hroubí (7cm). Mrtvé dřevo je popsáno průměrem (*diameter*), délkou (*length*) a stupněm rozkladu mrtvého dřeva (*state*).
- *litter* je tabulka záznamů materiálu opadu. Záznam tabulky obsahuje popis složky opadu (*litter_type*) a její zastoupení v opadu v procentech (*proportion*).



Obr. 9. Diagram tabulek databáze

6 APLIKACE PRO MIGRACI DAT

Součástí praktické části diplomové práce je návrh aplikace, která by usnadnila proces migrace dat ze zdrojových projektů měřených dat inventarizace lesů uložených v souborech databáze Access do navržené cílové struktury tabulek uložených v databázovém serveru PostgreSQL.

Aplikace pro migraci dat byla programována v programovacím jazyce C# v grafickém vývojovém prostředí Microsoft Visual Studio 2010 Express Edition. Pro přístup k souborům databáze Access aplikace využívá standardní rozhraní ODBC a komponenty .NET pro práci s tímto rozhraním přístupné prostřednictvím *System.Data.Odbc*. Pro přístup k databázi PostgreSQL aplikace využívá rozhraní *Npgsql*. Třídy tohoto rozhraní jsou k aplikaci připojeny z dynamicky linkovaných knihoven *Npgsql.dll* a *Mono.Security.dll*. Knihovna *Npgsql* byla použita zejména z důvodu snadného přístupu a práce s binárními soubory v rámci databáze PostgreSQL.

Kromě zmíněných knihoven pro práci s databázovým systémem aplikace používá také dynamicky linkovanou knihovnu *Ionic.Zip.dll*. Tato knihovna slouží ke kompresi a dekompresi souborů formátu zip. Projekty venkovního sběru dat jsou ukládány v rámci managementu inventarizačních ploch jako binární soubor v databázové tabulce. Tento soubor je uložen v komprimované podobě ve formátu souborů zip. Pro přístup k datům projektu venkovního sběru dat tedy aplikace musí nejdříve soubor obsahující zvolenou inventarizační plochu v databázi vyhledat, binární soubor z databáze přečíst a stáhnout. Stažený soubor je následně nutné dekomprimovat prostřednictvím funkcí externí knihovny a přečíst měřená data zvolené inventarizační plochy. Tato data jsou následně upravena podle definovaných změn a zpětně nahrána do cílové databáze v numerické podobě do příslušných datových tabulek.

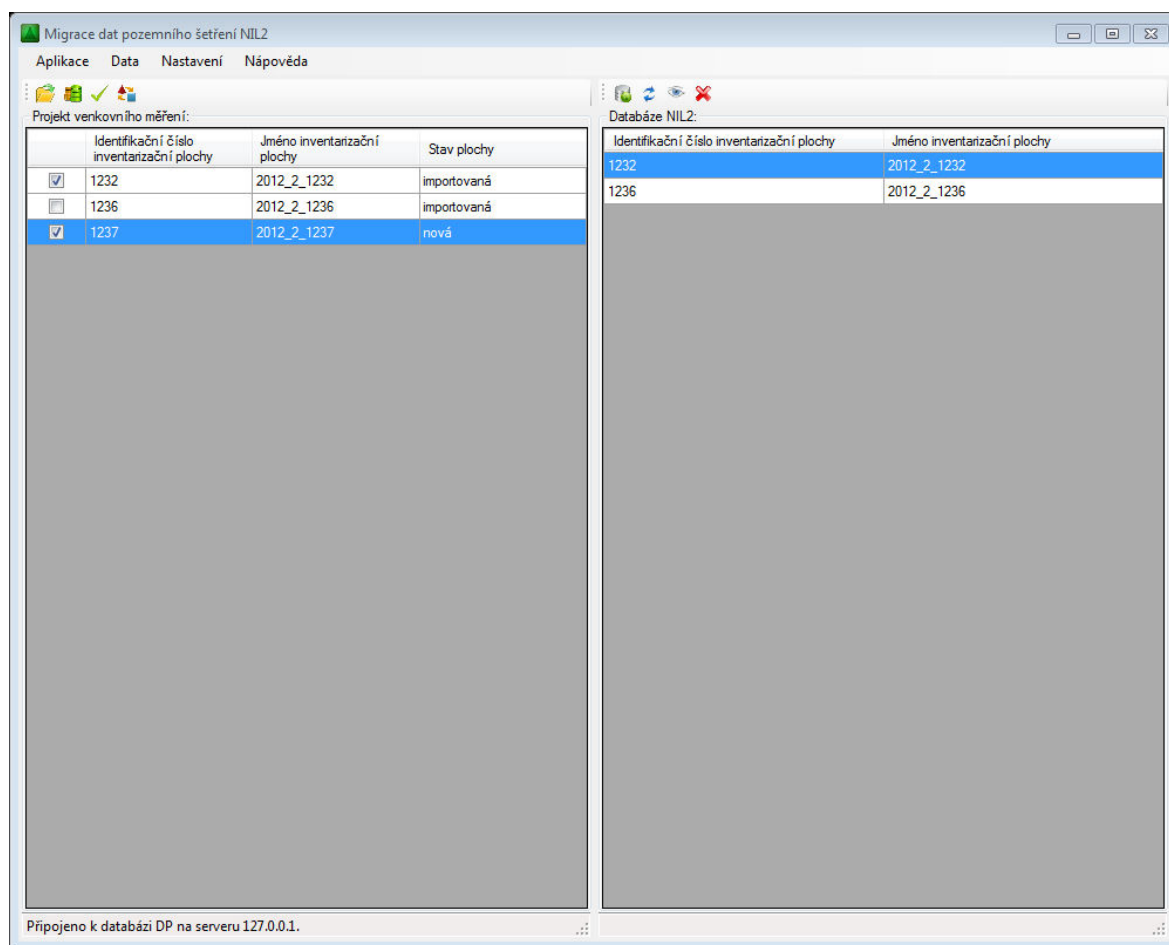
6.1 Grafické uživatelské rozhraní aplikace

Aplikace pro migraci dat je spouštěna prostřednictvím souboru *dp.exe*. Po spuštění aplikace se po chvíli zobrazí její hlavní formulář.

6.1.1 Hlavní formulář

Hlavní formulář (Obr. 10) obsahuje hlavní menu a pracovní plochu. Pracovní plocha je rozdělena na dvě poloviny, z nichž první v levé části formuláře je určena k zobrazování

seznamu inventarizačních ploch projektu venkovního měření a druhá v pravé části formuláře zobrazuje seznam ploch v cílové databázi inventarizace lesů. Každá z částí má svůj panel nástrojů s tlačítky a svůj stavový řádek. Stavový řádek informuje o aktuálně otevřeném souboru projektu venkovního měření nebo o stavu připojení aplikace k databázi PostgreSQL. Při prvním spuštění není otevřen žádný projekt venkovního měření a aplikace není připojena k cílové databázi, oba seznamy ploch jsou prázdné.



Obr. 10. Hlavní formulář aplikace pro migraci dat inventarizace lesů

6.1.2 Otevření lokálně uloženého souboru zdrojových dat

Aplikace umožňuje data načítat ze dvou zdrojů. Prvním možným zdrojem je soubor projektu venkovního měření lokálně uložený v souborovém systému počítače klienta.

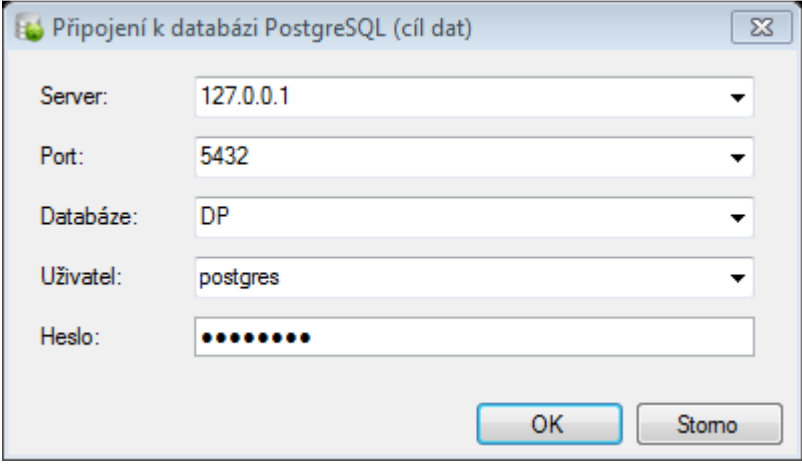
Pro otevření souboru projektu venkovního měření slouží první tlačítko panelu nástrojů *Otevřít zdroj dat*. Po kliknutí na tlačítko *Otevřít zdroj dat* aplikace zobrazí standardní dialog pro otevření souboru, který má nastaven výběr na mdb soubory databáze Access. V dialogu je nutné zvolit soubor mdb s daty venkovního měření inventarizace lesů a výběr

potvrdit tlačítkem *Otevřít*. U otvíraných souborů se očekává existence tabulky *plot*, ze které jsou načítána data o měřených inventarizačních plochách. Aplikace tedy neumožní import dat z libovolného souboru databáze Access a pokus o otevření jiného souboru než projektu venkovního měření inventarizace lesů končí ošetřenou výjimkou a zobrazením chybového hlášení. V případě, že vše proběhlo správně, aplikace do levého panelu hlavního formuláře vypíše tabulku se seznamem inventarizačních ploch obsažených ve zdrojovém projektu venkovního měření. V tabulce ploch jsou vypsána inventarizační čísla měřených ploch a u každého záznamu inventarizační plochy je doplněna informace o stavu této plochy v cílové databázi. Stav *nová* označuje plochy, které zatím v cílové databázi nemají importovány žádné záznamy. Stav *importovaná* označuje plochy, jejichž data již do databáze byla nahrána z téhož nebo jiných projektů venkovního šetření v rámci některého z předchozích importů. V případě, že aplikace není připojena k cílové databázi, pak informace o stavu ploch v cílové databázi chybí a všechny plochy jsou označeny jako nové.

Pro otevření lokálně uloženého souboru měřených dat je možné rovněž použít volbu z hlavního menu aplikace výběrem položky *Aplikace* a *Otevřít zdroj dat* nebo použít klávesovou zkratku *Ctrl+O*.

6.1.3 Připojení k cílové databázi

Před spuštěním přenosu dat ze zdrojového souboru do cílové databáze je nutné aplikaci k cílové databázi připojit. Nastavení připojení je možné provést kliknutím na tlačítko *Připojit k cílové databázi*, které je umístěno jako první na panelu nástrojů pravé části hlavního formuláře sloužící k zobrazování výpisu ploch databáze inventarizace lesů. Aplikace následně zobrazí formulář *Připojení k databázi PostgreSQL* (Obr. 11). Na formuláři je nutné zadat všechny parametry pro připojení k databázovému serveru, které jsou potřebné pro sestavení celého připojovacího řetězce a vytvoření objektu třídy *NpgsqlConnection*. Databázový server je možné zadávat jeho IP adresou nebo jeho jménem. Druhé textové pole určuje TCP port databázového serveru. Databázový server PostgreSQL standardně využívá portu 5432, pokud správcem serveru nebylo určeno jinak. Třetí textové pole slouží ke specifikaci jména databáze. Aplikace se připojuje k databázi inventarizace lesů. Tato databáze musí obsahovat založené schéma pro ukládání dat inventarizačních ploch, získaných při venkovním měření, které je pojmenované *nfi_data* a obsahuje základní tabulku inventarizačních ploch *plot*.



Obr. 11. Formulář připojení k databázi PostgreSQL

6.1.4 Otevření souboru zdrojových dat ze systému managementu ploch

Druhou možností a pravděpodobně častěji využívanou variantou bude import zdrojových dat ze systému managementu inventarizačních ploch.

Management inventarizačních ploch je systém sloužící k verzování a správě projektů venkovního měření. Data managementu inventarizačních ploch jsou uložena ve svém samostatném databázovém schématu v rámci databáze inventarizace lesů na databázovém serveru PostgreSQL. Datovou vrstvu managementu tvoří skupina čtyř tabulek, které obsahují informace o projektu venkovního měření (tabulka *batch_metadata*), o plochách obsažených v projektu venkovního měření (tabulka *plot_metadata*), binární data projektů (tabulka *batch_binary_data*) a výsledky kontrol projektu (tabulka *plot_data_errors*). Přijaté projekty venkovního měření procházejí kontrolou pomocí externí aplikace a po kontrole jsou komprimovány do zip souborů, které jsou pomocí externí aplikace nahrány jako binární data do tabulky binárních dat projektů. Výsledky kontrol a registrace ploch a projektů jsou uloženy do tabulek metadat. V této formě se uchovávají všechny projekty, tedy projekty bezchybné, projekty u kterých byla nalezena chybně změřená data, projekty kontrolních měření a projekty opravené. Stejná inventarizační plocha může tedy být změřena vícekrát a uložena ve více projektech. Určení správné poslední validní verze dat inventarizační plochy a její vyhledání ve správném projektu potom může být velmi nesnadný úkol. Určení validní verze inventarizační plochy je zajištěno již na straně databáze pomocí triggerů. Při každé registraci nového projektu a jeho inventarizačních ploch trigger spouští funkce, které pro každou plochu na základě výsledků kontrol

a stavu dříve nahraných verzí inventarizační plochy nastaví její validitu a validitu celého projektu.

Otevření souboru zdrojových dat ze systému managementu inventarizačních ploch začíná připojením k databázovému systému PostgreSQL, na kterém je umístěna databáze inventarizace lesů se zavedeným systémem managementu inventarizačních ploch. Po kliknutí na tlačítko *Otevřít management projektů* v seznamu tlačítek levého panelu hlavního formuláře aplikace zobrazí formulář *Připojení k databázi PostgreSQL (zdroj dat)*.

Návrh formuláře je stejný jako při připojení k cílové databázi (Obr. 11). Uživatel tedy musí uvést jméno nebo IP adresu databázového serveru, TCP port, jméno databáze a musí se identifikovat platným uživatelským jménem a heslem.

Pro připojení aplikace k managementu inventarizačních ploch je možné také použít volbu *Aplikace a Otevřít management projektů...* v nabídce hlavního menu nebo klávesovou zkratku *F2*.

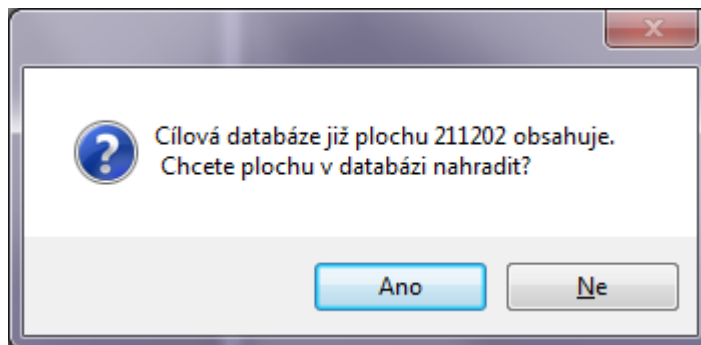
V případě importu dat ze systému managementu inventarizačních ploch aplikace udržuje dvě připojení k databázovému serveru PostgreSQL. Jedno připojení slouží jako zdroj dat a druhé jako cíl dat. Obecně tato připojení nemusí být shodná k téže databázi na tomtéž serveru. Aplikaci je tímto například možné použít ke stažení dat inventarizačních ploch, která jsou uložena v systému managementu inventarizačních ploch na vzdáleném serveru a jejich uložení do kopie databáze umístěné na PostgreSQL serveru na lokálním počítači.

6.1.5 Výběr ploch pro import do databáze

Po otevření souboru projektu venkovního měření nebo připojení aplikace k systému managementu inventarizačních ploch levý panel hlavního formuláře obsahuje seznam ploch projektu, jejichž data je možné importovat do cílové databáze. Uživatel volí plochu pro import zaškrtnutím příslušného zaškrťovacího políčka u vybrané inventarizační plochy. Při kliknutí na tlačítko *Vybrat všechny plochy* aplikace automaticky označí všechny plochy v otevřeném projektu jako plochy určené k importu do databáze.

Volit je možné plochy nové, ale také plochy importované. V případě, že byla zvolena plocha, u které již byla data importována v rámci některého z předchozích importů dat, bude uživatel na tuto skutečnost v průběhu importu dat znovu upozorněn dotazem (Obr. 12), zda nově importovaná data inventarizační plochy mají nahradit data dřívější. Při volbě *Ano*, aplikace nejdříve z tabulek databáze odstraní všechna stará data

inventarizační plochy a nahraje nová z aktuálně otevřeného projektu. Při volbě *Ne* bude inventarizační plocha ze seznamu ploch pro import vypuštěna a v databázi zůstanou zachována její původní dříve nahraná data.

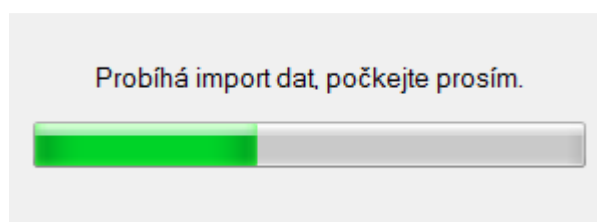


Obr. 12. Dialog nahrazení starých dat inventarizační plochy

6.1.6 Import numerických a textových dat ploch do databáze

Po výběru ploch na panelu zdrojového projektu venkovního měření je možné data vybraných ploch přesunout do cílové databáze inventarizace lesů. Import dat je spuštěn kliknutím na tlačítko *Importovat vybrané plochy do cílové databáze* umístěné v panelu nástrojů zdrojového projektu venkovního měření. Aplikace podle zadaného nastavení přenosu dat uloženého v konfiguračním souboru načte data jednotlivých tabulek zdrojového projektu venkovního měření. Pro jednotlivé záznamy tabulek podle předem definovaného mapování sloupců tabulek sestaví příkazy pro vložení do cílové databáze inventarizace lesů a tyto příkazy spustí nad cílovou databází.

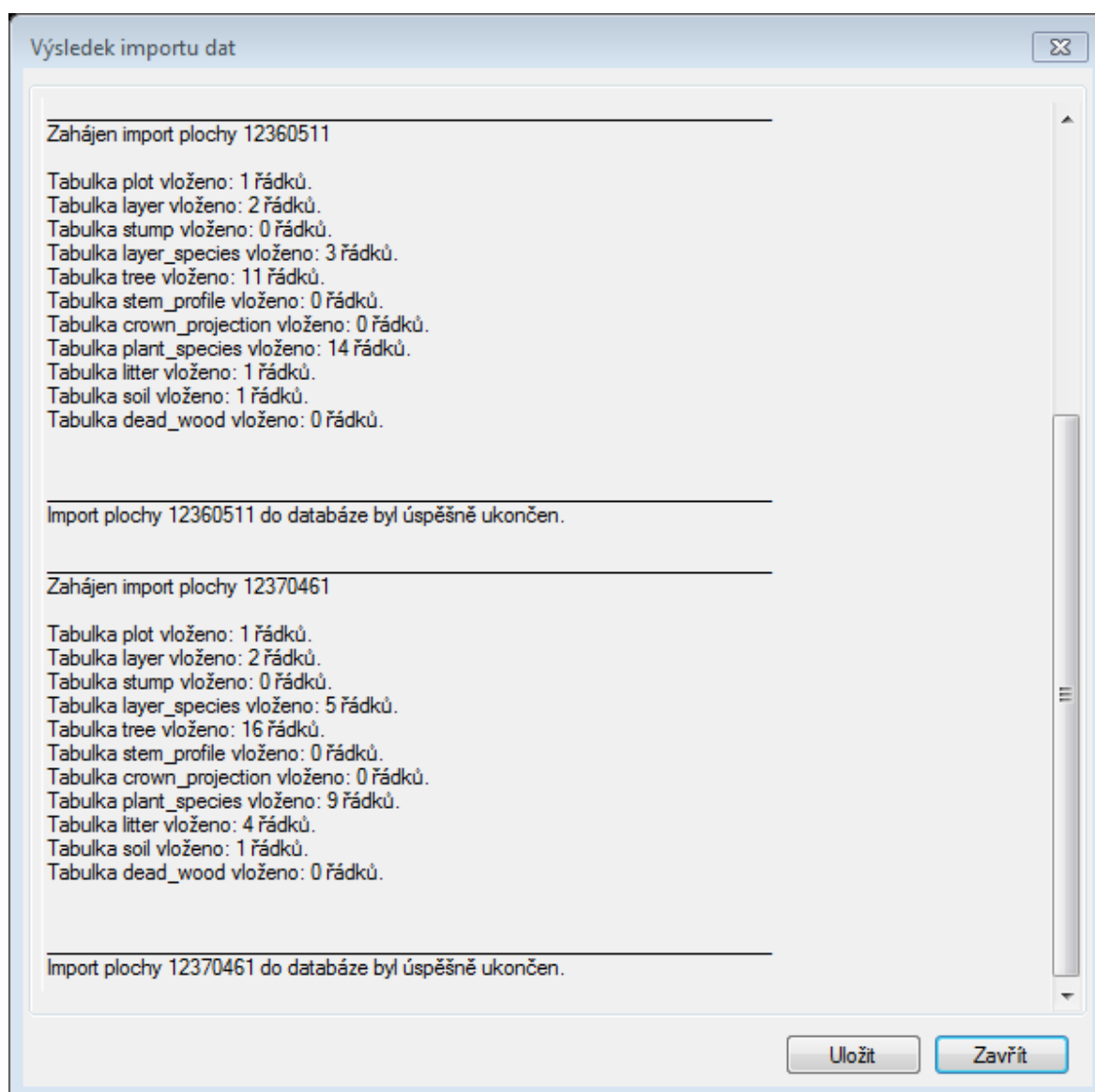
Průběh importu aplikace zobrazuje ve stavovém proužku zobrazeného formuláře. (Obr. 13)



Obr. 13. Formulář průběh importu dat

Při vládání dat do databáze je prováděna kontrola kvality měřených dat, kterou zajišťují jednotlivá omezení definovaná nad příslušnými tabulkami v databázi. V první řadě všechna importovaná data musí dodržet podmínky referenční a doménové integrity. Databáze během importu nedovolí vkládat sirotky, tedy záznamy bez vazby na rodičovský záznam v nadřazené tabulce. Příkladem může být vložení záznamu stromu, který neroste na žádné

inventarizační ploše. Cizí klíče tabulek rovněž umožňují kontrolovat atributy vázané na rozsah hodnot, který je definovaný číselníkem. Například tedy není možné vložit záznam stromu, pro který není v číselníku definován druh dřeviny. Řada atributů má definované omezení *Not Null*, požadující povinné uvedení hodnoty atributu. Prostřednictvím omezení *Check* byla v databázi rovněž implementována řada logických kontrol, definující rozsahy povolených hodnot a provádějící kontrolu, za kterých podmínek záznam nemusí být vyplněn.



Obr. 14. Formulář výsledek importu dat

Provedení importu dat jedné inventarizační plochy ze všech vybraných inventarizačních ploch probíhá v jedné transakci. Pokud tedy některé z definovaných omezení najde záznam, který není validní a není možné ho do příslušné databázové tabulky zapsat

je vyvolána výjimka. Celá transakce jedné inventarizační plochy bude zastavena a je proveden její *Rollback*. V tomto případě žádná změna v databázi nebude provedena. Chybný záznam spolu s omezením, které bylo porušeno, je vypsán v souboru logu a obsluha aplikace musí provést opravu tohoto záznamu a import chybné inventarizační plochy opakovat.

V případě, že import celé dávky proběhl v pořádku, je proveden *Commit* transakce a změny jsou zapsány v databázi. Aplikace zobrazí formulář *Výsledek importu dat* (Obr. 14) s uvedeným seznamem importovaných ploch a se seznamem tabulek s uvedenými počty záznamů, které byly do příslušných datových tabulek zapsány.

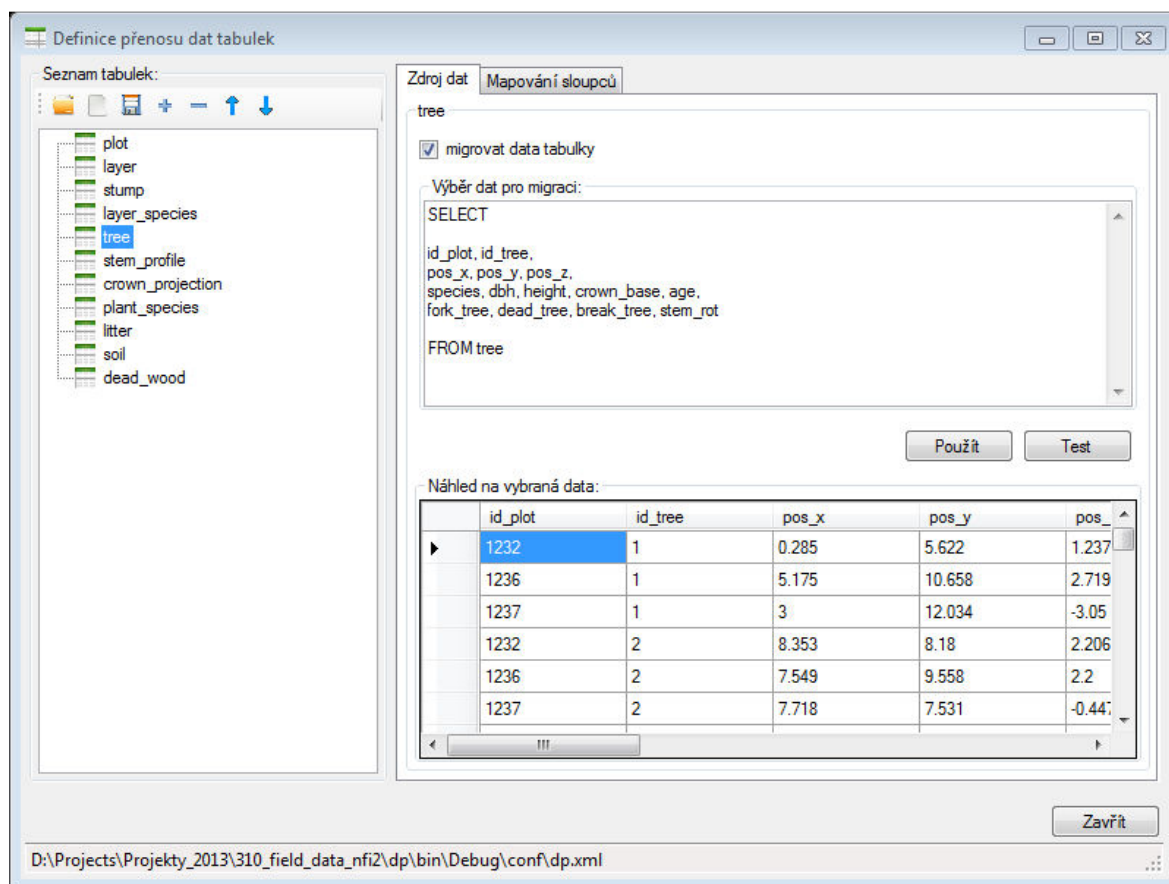
Formulář *Výsledek importu dat* umožní uživateli výsledek průběhu importu dat uložit do textového souboru.

6.1.7 Smazání vybrané plochy v cílové databázi

Seznam ploch, které byly importovány do databáze PostgreSQL při dříve provedených importech je zobrazen v tabulce v pravé části hlavního formuláře. Tento seznam je možné aktualizovat kliknutím na tlačítko *Načíst seznam ploch cílové databáze* na panelu nástrojů zobrazeném nad tabulkou se seznamem importovaných ploch. Tabulka se seznamem importovaných ploch umožňuje zvolit jeden řádek představující importovanou plochu. Kliknutím na tlačítko *Smazat plochu v cílové databázi* budou všechna data zvolené inventarizační plochy z databáze odstraněna.

6.1.8 Nastavení přenosu dat

Aplikace musí kromě provedení vlastního přenosu dat ze zdrojového souboru projektu měření v databázi Access do cílové databáze umožnit také definovat, jaká data ze zdrojového souboru budou přenášena a do kterých tabulek tato data budou ukládána. K tomuto účelu je určen v aplikaci formulář *Definice importu dat tabulek* (Obr. 15). Uživatel může zobrazit tento formulář výběrem položky *Nastavení a Konfigurace přenosu dat* v menu hlavního formuláře aplikace nebo použitím klávesové zkratky *F9*.



Obr. 15. Formulář definice importu dat tabulek

Formulář umožní načíst seznam tabulek v cílové databázi PostgreSQL, do kterých požadujeme nahrávání dat, a pro každou tuto tabulku v cílové databázi definovat zdroj dat. Provedené nastavení přenosu dat je možné uložit do konfiguračních souborů ve formátu xml a opakovaně je používat.

V levé části formuláře *Definice přenosu dat tabulek* je zobrazen seznam tabulek, pro které konfigurační soubor obsahuje definici přenosu dat. V horní části seznamu je zobrazen panel nástrojů s tlačítky.

První tlačítko *Otevřít konfigurační soubor* zobrazí standardní dialog pro otevření souboru, které umožní vyhledat a zvolit dříve vytvořený konfigurační soubor pro přenos dat. Aplikace má vždy otevřen jeden konfigurační soubor jako aktivní, který používá při spuštění migrace dat do cílové databáze. Jméno otevřeného aktivního konfiguračního souboru je zobrazeno v dolní části formuláře na jeho stavovém řádku.

Zatímco schéma cílové databáze, které slouží k integraci dat z mnoha zdrojových projektů venkovního měření, je pevně navrženo a téměř se nemění, ve zdrojových projektech docházelo ke změnám mezi sezónami měření, které byly způsobeny dodatečnými

úpravami metodiky měření mezi jednotlivými sezónami. To v podstatě znamená, že zdrojové soubory mohou mít různou strukturu tabulek mezi jednotlivými sezónami. Aplikace proto musí umožnit vytvořit si konfigurační soubory pro jednotlivé verze zdrojových souborů. Při kliknutí na druhé tlačítko *Nový konfigurační soubor* aplikace zobrazí formulář pro zadání jména nového souboru (Obr. 16) a vytvoří nový prázdný konfigurační soubor v adresáři *conf*.

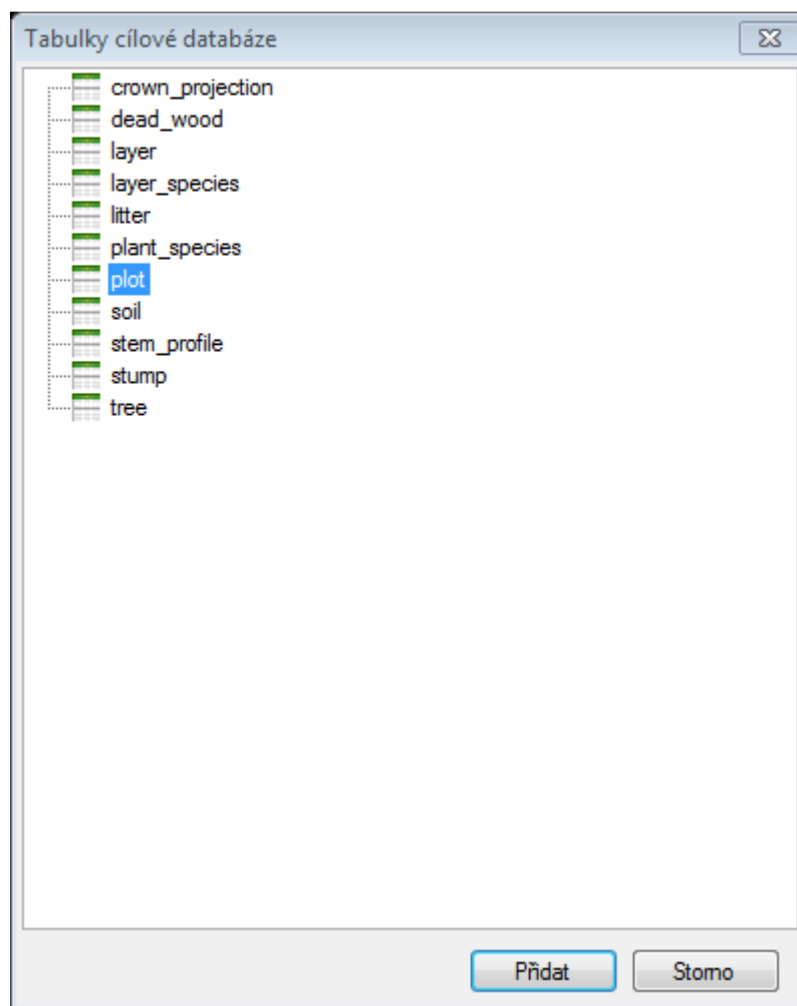
Obr. 16. Formulář nový konfigurační soubor

Otevřený konfigurační soubor je možné upravovat. Třetí tlačítko *Uložit konfigurační soubor* na panelu nástrojů provedené změny запиše do otevřeného konfiguračního souboru.

6.1.9 Výběr cílových tabulek pro import dat

Seznam tabulek cílové databáze, do kterých data budou nahrávána, je možné upravovat pomocí tlačítek *Přidat nebo odebrat tabulku z konfiguračního souboru*. Data budou migrována pouze do těch tabulek, pro které je v konfiguračním souboru definován přenos dat. Pokud je aplikace připojena k cílové databázi, kliknutím na tlačítko *Přidat tabulku do konfiguračního souboru* se zobrazí formulář *Tabulky cílové databáze* (Obr. 17). Výběrem jména tabulky a kliknutím na tlačítko *Přidat* je tabulka přidána do seznamu tabulek v konfiguračním souboru přenosu dat.

Data tabulek jsou migrována v pořadí, v jakém jsou tabulky uvedeny v seznamu v levé části formuláře *Definice přenosu dat tabulek*. Pořadí tabulek je možné upravovat kliknutím na tlačítka s ikonami šipek. Tato tlačítka posouvají vybranou tabulku v seznamu tabulek o jednu položku nahoru nebo dolů.



Obr. 17. Formulář tabulky cílové databáze

6.1.10 Definice zdroje dat pro cílovou tabulku

Pro každou nově přidanou tabulku je nutné definovat zdroj dat. Zdroj dat je definován prostřednictvím SQL dotazu, který je spuštěn v databázi Access obsahující zdrojová měřená data. Výsledek SQL dotazu je poté ukládán do cílové databáze a vybrané tabulky. Pro každou cílovou tabulku je nutné definovat SQL dotaz pro výběr dat ze zdrojového souboru a tento SQL dotaz uložit v konfiguračním souboru. Výběrový SQL dotaz je uživatelem zapsán do textového pole *Výběr dat pro migraci* na záložce *Zdroj dat* formuláře *Definice přenosu dat tabulek* (Obr 15). Validitu výběrového dotazu je nutné ověřit kliknutím na tlačítko *Test*. Pokud je aplikace připojena ke zdrojovému souboru databáze Access a zapsaný dotaz je správný, pak výsledek dotazu je zobrazen v tabulce *Náhled na vybraná data*. V případě, že výběrový dotaz obsahuje syntaktické chyby, bude zobrazena chybová hláška.

6.1.11 Mapování sloupců mezi zdrojovou a cílovou datovou tabulkou

Poté co byl definován výběrový dotaz, je nutné definovat mapování sloupců. Mapováním sloupců je určen datový sloupec výsledku výběrového dotazu obsahující zdrojová data a tento sloupec je přiřazen datovému sloupci v cílové databázové tabulce, do kterého zdrojová data sloupce budou uložena. Mapování sloupců je nutné definovat pro každou tabulku cílové databáze, do které data budou ukládána. Mapování sloupců je definováno na formuláři *Definice přenosu dat tabulek* v záložce *Mapování sloupců* (Obr. 18). Pro vybranou tabulku v seznamu v levé části formuláře je zobrazen seznam jejích datových sloupců. Pro každý sloupec je zobrazeno výběrové pole obsahující datové sloupce určené výběrovým SQL dotazem ze zdrojových dat. Ve výběrovém poli uživatel volí jméno datového sloupce ve zdroji dat a přiřazuje ho k datovému sloupci cílové tabulky. Pokud zdroj dat neobsahuje data pro cílový datový sloupec, výběrové pole obsahuje položku *not defined*. Při nastavení této položky budou do cílového datového sloupce při migraci dat aplikací dosazovány hodnoty NULL.

Definice přenosu dat tabulek

Seznam tabulek:

- plot
- layer
- stump
- layer_species
- tree
- stem_profile
- crown_projection
- plant_species
- litter
- soil
- dead_wood

Zdroj dat Mapování sloupců

tree

Určení párů cílový a zdrojový sloupec:

Sloupec databáze NIL2	Zdroj dat pro sloupec
Identifikační číslo plochy	id_plot
Identifikační číslo stromu	id_tree
Souřadnice x stromu vzhledem ke středu pl...	pos_x
Souřadnice y stromu vzhledem ke středu pl...	pos_y
Souřadnice z stromu vzhledem ke středu pl...	pos_z
Dřevina	species
Výčetní tloušťka v mm	dbh
Výška stromu v m	height
Výška koruny v m	crown_base
Věk	age
Rozdvojení kmene	fork_tree
Souš	dead_tree
Zlomený kmen	break_tree
Hniloba kmen	stem_rot

Použít

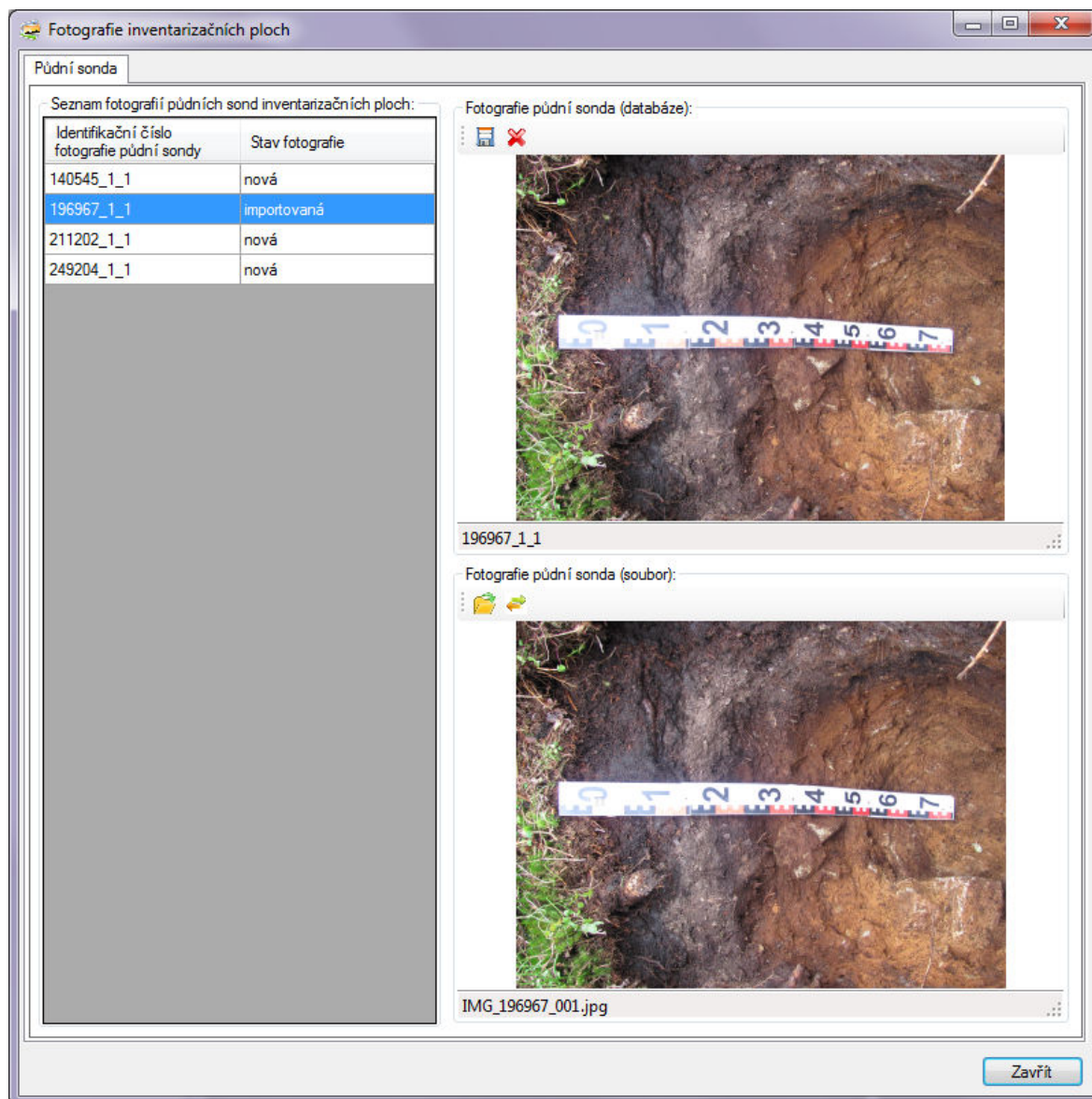
Zavřít

D:\Projects\Projekty_2013\310_field_data_nfi2\dp\bin\Debug\conf\dp.xml

Obr. 18. Formulář mapování sloupců

6.1.12 Import a export binárních souborů fotografií půdních sond

Kromě části numerických dat jsou v rámci měření na plochách pořizovány dokumentační fotografie, které je nutné ukládat zároveň s měřenými daty. Na plochách, kde byl prováděn půdní průzkum, byly pořízeny fotografie půdních sond. Fotografie půdních sond jsou v databázi ukládány ve formě *large object*. V tabulce fotografií půdních sond je pak uložen identifikátor *oid* pro daný *large object*.



Obr. 19. Formulář fotografie inventarizačních ploch

Pro import fotografií je určen formulář *Fotografie inventarizačních ploch* (Obr. 19). Tabulka v levé části formuláře zobrazuje seznam inventarizačních ploch, které mají v databázi již z dřívějšího importu nahraná měřená data. Pokud fotografie pro danou inventarizační plochu chybí, je plocha ve sloupci *Stav fotografie* označena jako *nová*.

Pokud fotografie již byla nahrána, je fotografie označena jako *importovaná*. Při kliknutí v tabulce seznamu inventarizačních ploch na plochu, která má fotografii importovanou, aplikace načte v databázi uloženou fotografii a tuto fotografii zobrazí v komponentě *PictureBox* v horní polovině formuláře.

V databázi uloženou fotografii je možné stáhnout a uložit do souboru kliknutím na první tlačítko panelu nástrojů *Uložit fotografii do souboru*. Po kliknutí na toto tlačítko je zobrazen standardní dialog pro uložení souboru a je očekáváno zadání jména souboru a cesty, kam má být fotografie uložena.

Druhé tlačítko panelu nástrojů *Odstranit fotografii z databáze*, provede smazání fotografie pro vybranou inventarizační plochu z databáze. Aplikace provede smazání fotografie ve dvou krocích. Nejdříve získá identifikátor fotografie *oid* z tabulky fotografií inventarizačních ploch a poté odstraní binární data fotografie z tabulky katalogu *pg_large_object*. Poté je teprve odstraněn vlastní záznam fotografie z tabulky fotografií inventarizačních ploch. Pouhé vymazání záznamu z tabulky fotografií nestačí, protože v databázi zůstávají stále uložena binární data fotografií, na které již nevedou žádné odkazy.

Dolní polovina formuláře slouží k zobrazení fotografie uložené v souboru, který bude nahrán do databáze. První tlačítko panelu nástrojů *Otevřít fotografii ze souboru* zobrazí standardní dialog pro otevření souboru s fotografií. Po určení souboru je fotografie pro import zobrazena v komponentě *PictureBox* v dolní části formuláře.

Otevřenou fotografii je poté možné importovat do databáze kliknutím na druhé tlačítko panelu nástrojů *Nahrát fotografii do databáze*. Aplikace poté vytvoří nový *large object* a získá jeho identifikátor *oid*. Binární data fotografie zapíše do tabulky *pg_large_object* a vytvoří nový záznam v tabulce fotografií inventarizačních ploch, do které uloží identifikátor jako odkaz na binární data fotografie.

6.1.13 Náповěda k aplikaci

K aplikaci pro migraci dat inventarizačních ploch byla vytvořena nápověda. Pro vytvoření nápovědy byla použita aplikace *HTML Help Workshop*. Jednotlivá témata nápovědy jsou tvořena samostatnými html soubory, které byly aplikací *HTML Help Workshop* kompilovány do binárního souboru s příponou *chm*. Soubor kompilované nápovědy je součástí aplikace pro migraci dat. Uživatel aplikace může obsah souboru zobrazit

kliknutím na položku *Nápověda* a výběrem *Zobrazit nápovědu* v nabídce hlavního formuláře aplikace. Soubor nápovědy je možné také otevřít při práci s aplikací za použití klávesové zkratky *F1*.

6.1.14 Ukončení práce s aplikací

Aplikace pro migraci dat je ukončena zavřením hlavního formuláře. Aplikaci je možné také uzavřít použitím položky *Konec* v nabídce *Aplikace* hlavního formuláře nebo použitím klávesové zkratky *Ctrl+Q*.

6.2 Programová dokumentace aplikace

Část programové dokumentace aplikace je věnována popisu hlavních tříd, jejich vlastností a metod, které tvoří jádro aplikace pro migraci dat.

6.2.1 Třída *PlotList*

Třída je určena pro práci se seznamy inventarizačních ploch. Obecná třída obsahuje společné vlastnosti a metody pro třídy seznamů ploch pro import z databáze Access a seznamů importovaných ploch v databázi PostgreSQL.

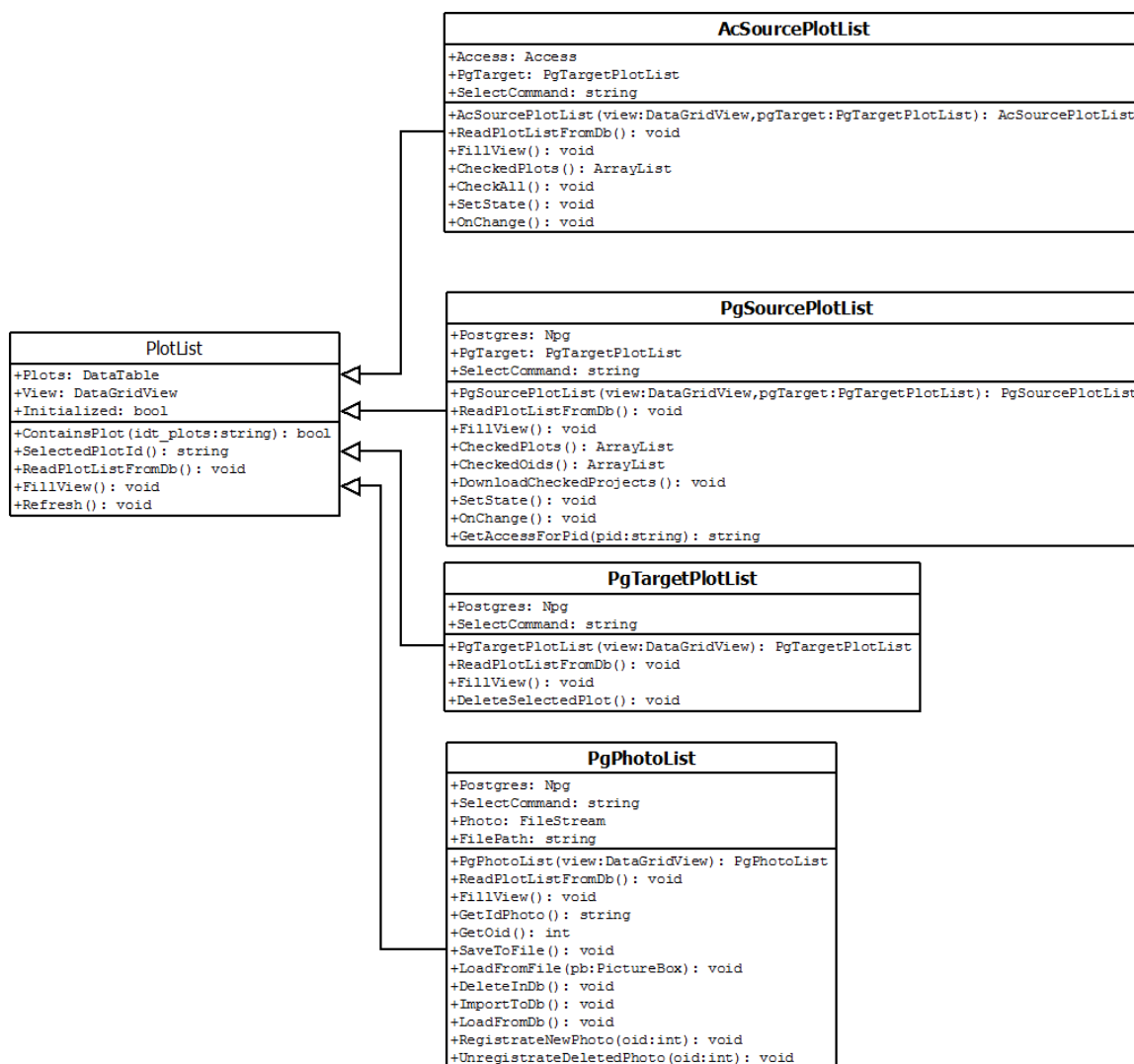
Vlastnosti:

- *Plots* – vlastnost typu *DataTable*, obsahuje tabulku seznamu ploch načtenou z databáze.
- *View* – vlastnost typu *DataGridView*, obsahuje odkaz na komponentu hlavního formuláře, do které bude seznam ploch vypisován.
- *Initialized* – vlastnost datového typu *boolean*, obsahuje informaci o tom, zda seznam inventarizačních ploch byl z databáze načten.

Metody:

- *ContainsPlot* - metoda přijímá v argumentu identifikátor inventarizační plochy a vrací logickou hodnotu, která určuje, zda seznam inventarizačních ploch obsahuje plochu určenou jejím identifikátorem.
- *SelectedPlotId* – metoda vrací textový řetězec s identifikátorem inventarizační plochy, která byla v seznamu vybrána.
- *ReadPlotListFromDb* – abstraktní metoda, která je implementována až v potomcích třídy *PlotList*.

- *FillView* – abstraktní metoda, která je implementována až v pomocích třídy *PlotList*.
- *Refresh* – metoda je určena k znovu načtení seznamu ploch v databázi. Aktualizuje zobrazení přehled ploch na hlavním formuláři.



Obr. 20. Diagram tříd – seznamy inventarizačních ploch

6.2.2 Třída AcSourcePlotList

Tato třída je určena k práci se seznamem inventarizačních ploch, které jsou obsažené ve zdrojovém souboru databáze Access. Třída je potomkem třídy *PlotList*.

Vlastnosti:

- *Access* – vlastnost obsahuje odkaz na objekt připojení ke zdrojové databázi.

- *PgTarget* – vlastnost obsahuje odkaz na objekt se seznamem importovaných ploch v cílové databázi PostgreSQL.
- *SelectCommand* – vlastnost obsahuje textový řetězec s SQL příkazem pro výběr seznamu ploch ve zdrojové databázi, které mají být importovány

Metody:

- *AcSourcePlotList* – konstruktor objektu, provádí inicializaci proměnných nového objektu.
- *ReadPlotListFromDb* – metoda načítá seznam inventarizačních ploch, které jsou obsažené v tabulce inventarizačních ploch uložené ve zdrojovém souboru databáze Access.
- *FillView* – metoda zobrazuje seznam inventarizačních ploch zdrojového souboru databáze Access v komponentě *DataGridView* hlavního formuláře. K zobrazenému seznamu přidává doplňující informace, zda je daná plocha vybrána k importu a zda již tato plocha nebyla importovaná dříve a není obsažena v cílové databázi PostgreSQL.
- *CheckedPlots* – metoda vrací kolekci *ArrayList* se seznamem identifikátorů inventarizačních ploch, které uživatel v seznamu ploch obsažených ve zdrojovém souboru Access, zaškrtnul a vybral je pro import od cílové databáze PostgreSQL.
- *CheckAll* – metoda označí nebo zruší označení všech inventarizačních ploch v seznamu ploch zdrojového souboru, určených k importu.
- *SetState* – metoda porovná seznam ploch zdrojového souboru a seznam ploch v cílové databázi. Pokud plocha již je v cílové databázi obsažena, označí plochu v seznamu jako importovanou. Pokud se plocha v cílové databázi nenachází, označí ji jako novou.
- *OnChange* – obsluha události. Při změně v seznámech inventarizačních ploch v cílové databázi je vyvolána obsluha události *OnChange*, která znovu nastaví stav ploch v seznamu.

6.2.3 Třída *PgSourcePlotList*

Tato třída je určena k práci se seznamem inventarizačních ploch, které jsou obsažené ve zdrojových projektech dat evidovaných v systému managementu inventarizačních ploch v databázových tabulkách systému PostgreSQL. Třída je potomkem třídy *PlotList*.

Vlastnosti:

- *Postgres* – vlastnost obsahuje odkaz na objekt připojení ke zdrojové databázi PostgreSQL.
- *PgTarget* – vlastnost obsahuje odkaz na objekt se seznamem importovaných ploch v cílové databázi PostgreSQL.
- *SelectCommand* – vlastnost obsahuje textový řetězec s SQL příkazem pro výběr seznamu ploch ve zdrojové databázi PostgreSQL, kde jsou uloženy tabulky systému managementu inventarizačních ploch.

Metody:

- *PgSourcePlotList* – konstruktor objektu, provádí inicializaci proměnných nového objektu.
- *ReadPlotListFromDb* – metoda načítá seznam inventarizačních ploch, jejichž projekty byly nahrány v systému managementu inventarizačních ploch.
- *FillView* – metoda zobrazuje seznam inventarizačních ploch databáze managementu inventarizačních ploch v komponentě *DataGridView* hlavního formuláře. K zobrazenému seznamu přidává doplňující informace, zda je daná plocha vybrána k importu a zda již tato plocha nebyla importovaná dříve a není obsažena v cílové databázi PostgreSQL.
- *CheckedPlots* – metoda vrací kolekci *ArrayList* se seznamem identifikátorů inventarizačních ploch, které uživatel v seznamu ploch obsažených v systému managementu inventarizačních ploch, zaškrtnul a vybral je pro import od cílové databáze PostgreSQL.
- *CheckedOids* – metoda vrací objekt třídy *DataTable*. Datová tabulka obsahuje seznam identifikačních čísel a jmen projektů, které obsahují vybrané inventarizační plochy.
- *DownloadCheckedProjects* – metoda z databáze načte všechny soubory projektů, které obsahují zvolené inventarizační plochy a tyto soubory uloží v lokálním adresáři *data*.
- *GetAccessForPid* – metoda vrací textový řetězec s cestou k projektu měření dat obsahující zvolenou plochu určenou jejím identifikačním číslem.
- *SetState* - metoda porovná seznam ploch v systému managementu inventarizačních ploch a seznam ploch v cílové databázi. Pokud plocha již je v cílové databázi

obsažena, označí plochu v seznamu jako importovanou, pokud se plocha v cílové databázi nenachází, označí ji jako novou.

- *OnChange* – obsluha události. Při změně v seznamech inventarizačních ploch v cílové databázi je vyvolána obsluha události *OnChange*, která znovu nastaví stav ploch v seznamu.

6.2.4 Třída *PgTargetPlotList*

Třída slouží k práci se seznamem inventarizačních ploch, jejichž data byla již dříve nahrána do cílové databáze. Třída je potomkem třídy *PlotList*.

Vlastnosti:

- *Postgres* – vlastnost obsahuje odkaz na objekt připojení k cílové databázi PostgreSQL.
- *SelectCommand* – vlastnost obsahuje textový řetězec s SQL příkazem pro výběr seznamu ploch obsažených v cílové databázi PostgreSQL.

Metody:

- *PgTargetPlotList* – konstruktor objektu, provádí inicializaci proměnných nového objektu.
- *ReadPlotListFromDb* – metoda načítá seznam inventarizačních ploch, které jsou již obsaženy v cílové databázi PostgreSQL.
- *FillView* – metoda zobrazuje seznam inventarizačních ploch v cílové databázi na komponentě *DataGridView* hlavního formuláře.
- *DeleteSelectedPlot* – metoda odstraňuje všechna data vybrané inventarizační plochy z databáze.

6.2.5 Třída *PgPhotoList*

Třída je určena ke správě a nahrávání fotografií inventarizačních ploch do cílové databáze.

Vlastnosti:

- *Postgres* – vlastnost obsahuje odkaz na objekt připojení k cílové databázi PostgreSQL.
- *SelectCommand* – textový řetězec SQL příkazu pro výběr seznamu fotografií inventarizačních ploch.

- *Photo* – objekt třídy *FileStream*, obsahuje binární data fotografie načtené ze souboru.
- *FilePath* - cesta k souboru s načítanou fotografií.

Metody:

- *PgPhotoList* - konstruktor objektu seznamu fotografií provádí inicializaci všech proměnných nového objektu.
- *ReadPlotListFromDb* - metoda načítá seznam inventarizačních ploch, pro které se očekává nahrávání fotografie inventarizační plochy.
- *FillView* - metoda zobrazuje seznam inventarizačních ploch v komponentě *DataGridView*. K zobrazenému seznamu přidává doplňující informace, zda daná plocha v databázi již má uloženou fotografii nebo se očekává uložení fotografie.
- *GetIdPhoto* - metoda vrací identifikační číslo fotografie, které bylo uživatelem vybráno v seznamu.
- *GetOid* - metoda vrací identifikační číslo objektu binárních dat fotografie v databázi, které bylo uživatelem vybráno v seznamu.
- *SaveToFile* - metoda načte binární data vybrané fotografie z databáze a uloží ji do zvoleného souboru.
- *LoadFromFile* - metoda načte data zvoleného souboru fotografie a zobrazí ji v komponentě *PictureBox* formuláře.
- *DeleteInDb* - metoda odstraňuje vybranou fotografii z databáze. Metoda vymaže příslušný *large object* fotografie z katalogu a odstraní záznam fotografie v datové tabulce.
- *ImportToDb* - metoda importuje data vybrané fotografie ze souboru. Metoda uloží fotografie ve formě *large object* do katalogu a vytvoří záznam o fotografii v datové tabulce.
- *LoadFromDb* - metoda načítá data zvolené fotografie z databáze a zobrazí ji v komponentě *PictureBox* formuláře.
- *RegistrateNewPhoto* - metoda vytváří nový záznam o fotografii v datové tabulce.
- *UnregistrateDeletedPhoto* - metoda ruší záznam vybrané fotografie v datové tabulce.

6.2.6 Třída ETLDefinition

Třída obsahuje informace o nastavení přenosu dat mezi zdrojovým a cílovým systémem. Tyto informace umožňuje načítat a ukládat do konfiguračního souboru. Informace o nastavení přenosu dat poskytuje ostatním třídám aplikace.

Vlastnosti:

- *DsETL* – objekt třídy *DataSet* obsahuje dvě tabulky s názvy *Tables* a *Columns*.
- *Tables* – objekt třídy *DataTable*. Ve formě datové tabulky obsahuje zapsané informace o databázových tabulkách, do kterých budou přenášena data importována. Objekt *DataTable* obsahuje sloupce se jménem (*table*) a schématem (*schema*) cílové databázové tabulky, sloupec *source* s definicí výběrového SQL dotazu pro získání dat pro naplnění databázové tabulky, sloupec *import* s logickou hodnotou definující, zda data tabulky budou migrována při spuštění importu a sloupec *order* určující pořadí v jakém budou data do cílové tabulky migrována.
- *Columns* – objekt třídy *DataTable*. Ve formě datové tabulky uchovává informace o mapování datových sloupců pro jednotlivé databázové tabulky. Tabulka obsahuje sloupce *schema* a *table* identifikující databázovou tabulku, sloupec *target_caption* obsahující popis sloupce v cílové databázové tabulce zobrazovaný na formuláři pro definici mapování sloupců. A sloupce *target* a *source*, které obsahují jména sloupců cílové tabulky a jméno sloupce ve výsledku výběrového SQL dotazu z dat zdrojového systému.
- *FileName* – vlastnost obsahuje textový řetězec s cestou ke konfiguračnímu souboru s definicemi migrace dat.
- *Changed* – logická hodnota určující, zda nastavení migrace dat bylo změněno a nebylo provedeno uložení změn do konfiguračního souboru.

Metody:

- *ETLDefinition* – konstruktor objektu třídy *ETLDefinition*.
- *CreateNewTables* - metoda vrací prázdný objekt třídy *DataTable* pro uložení informací o datových tabulkách.
- *CreateNewColumns* - metoda vrací prázdný objekt třídy *DataTable* pro uložení informací o párech datových sloupců.
- *LoadFromFile* – metoda nahrává definici přenosu dat z konfiguračního souboru ve formátu xml.

- *SaveToFile* – metoda ukládá definici přenosu dat do konfiguračního souboru ve formátu xml.
- *GetTableList* - metoda vrací kolekci *ArrayList* obsahující seznam jmen datových tabulek obsažených v konfiguračním souboru.
- *ContainsTable* – metoda ověřuje existenci zadané datové tabulky v seznamu tabulek v konfiguračním souboru.
- *DeleteTable* - metoda odstraňuje zadanou datovou tabulku ze seznamu tabulek v konfiguračním souboru.
- *AddTable* – metoda přidává záznam nové datové tabulky do seznamu tabulek v konfiguračním souboru.
- *AddColumnPair* - metoda přidává záznam nového páru datových sloupců do seznamu sloupců v konfiguračním souboru.
- *GetSourceSql* – metoda vrací textový řetězec s definicí výběrového SQL dotazu pro načtení dat datové tabulky ze zdrojového systému.
- *SetSourceSql* – metoda pro zvolenou tabulku ukládá do konfiguračního souboru její novou definici výběrového SQL dotazu pro načítání dat ze zdrojového systému.
- *GetTableImportInfo* - metoda vrací logickou hodnotu určující, zda data zvolené tabulky budou migrována.
- *EnableTableForImport* - metoda označuje vybranou datovou tabulku, jako tabulku pro import dat.
- *DisableTableForImport* - metoda označuje vybranou datovou tabulku jako tabulku, do které data nebudou nahrávána.
- *GetTableOrder* - metoda vrací pořadové číslo tabulky.
- *SetTableOrder* - metoda nastavuje pořadové číslo tabulky.
- *GetColumnMappings* - metoda vrací pole objektů třídy *DataRow*, které obsahují informace o párech zdrojového a cílového sloupce pro zvolenou datovou tabulku.
- *SetColumnMappings* - metoda pro zvolenou cílovou tabulku a datový sloupec nastavuje jméno zdrojového datového sloupce.
- *GetTablesForImport* - metoda vrací seznam tabulek ve formě objektu třídy *ArrayList*. Seznam tabulek je seřazen podle pořadí, v jakém data tabulek budou importovány.
- *GetTargetColumnList* - metoda vrací seznam datových sloupců cílové tabulky jako objekt třídy *ArrayList*.

- *GetTargetColumnString* - metoda vrací seznam datových sloupců cílové tabulky ve formě textového řetězce. Jména datových sloupců jsou oddělena znakem čárky.
- *GetColumnPairs* - vrací objekt třídy *Dictionary* obsahující páry zdrojový a cílový datový sloupec.

6.2.7 Třída *DataMigration*

Hlavní třída celé aplikace. Třída je určena k načtení dat ze zdrojové databáze a nahrání dat do cílové databáze.

Vlastnosti:

- *AcSource* - vlastnost obsahuje odkaz na objekt třídy *AcSourcePlotList* se seznamem inventarizačních ploch pro import ve zdrojovém souboru databáze Access.
- *PgSource* - vlastnost obsahuje odkaz na objekt třídy *PgSourcePlotList* se seznamem inventarizačních ploch pro import ve zdrojovém systému, kterým je systém managementu inventarizačních ploch v databázi PostgreSQL.
- *PgTarget* - vlastnost obsahuje odkaz na objekt třídy *PgTargetPlotList*, který obsahuje seznam importovaných ploch v cílové databázi.
- *PgPhotoList* - vlastnost obsahuje odkaz na objekt třídy *PgPhotoList*, který obsahuje seznam fotografií v cílové databázi.
- *Etl* - vlastnost obsahuje odkaz na objekt třídy *EtlDefinition*, který popisuje přenos dat datových tabulek mezi zdrojovým a cílovým systémem.
- *SourceType* - vlastnost, která určuje použitý zdroj dat.

Metody:

- *DataMigration* - konstruktor objektu. Vytváří nový objekt třídy *DataMigration* a všechny jeho součásti.
- *Execute* - metoda spouští přenos dat ze zdrojového systému do cílového.
- *ReplacePlots* - metoda odstraňuje ze seznamu inventarizačních ploch určených pro import inventarizační plochy, které jsou již v cílové databázi obsaženy a není požadováno jejich přepsání.
- *AccessToPostgres* - metoda provádí přenos dat ze zdrojového souboru databáze Access do cílové databáze PostgreSQL.
- *PostgresToPostgres* - metoda stáhne binární soubory zdrojových projektů z databáze PostgreSQL systému managementu inventarizačních ploch

do pracovního adresáře, stažené soubory dekomprimuje a data vybraných inventarizačních ploch z těchto projektů nahraje do cílové databáze PostgreSQL.

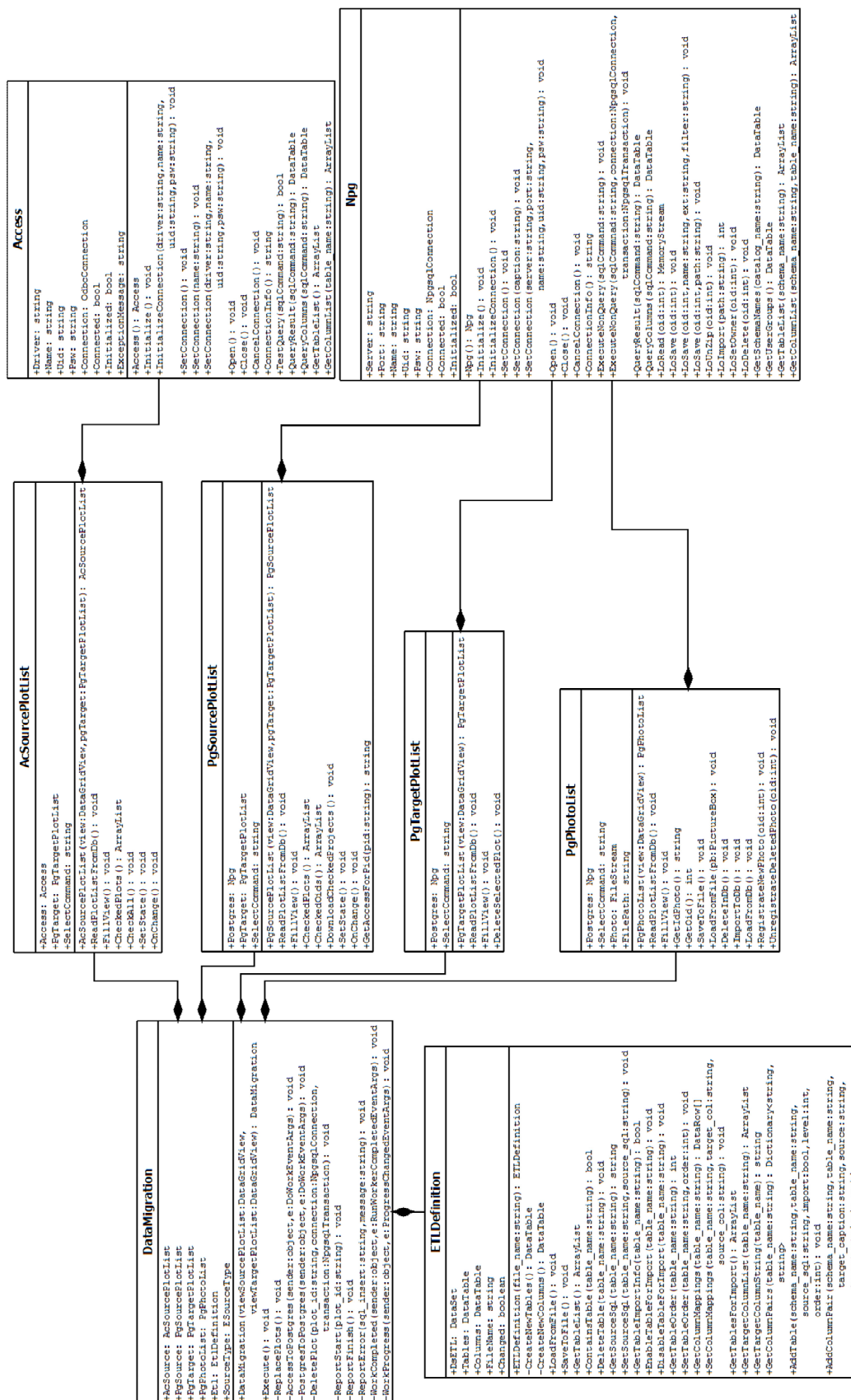
- *DeletePlot* - metoda vymaže inventarizační plochu z cílové databáze před jejím opětovným nahráním, pokud bylo uživatelem požadováno nahrazení dat inventarizační plochy.
- *ReportStart* - metoda zapisuje začátek přenosu dat do souboru logu.
- *ReportFinish* - metoda zapisuje úspěšné ukončení přenosu dat do souboru logu.
- *ReportError* - pokud při přenosu dat došlo k výjimce, metoda zapisuje hlášení o výjimce do souboru logu.
- *WorkCompleted* - metoda informuje ukončení průběhu importu dat.
- *WorkProgress* - metoda informuje o průběhu importu dat.

6.2.8 Třída Access

Třída Access je určena k práci s datovými soubory databáze Access. Třída používá připojení k souborům Access prostřednictvím rozhraní ODBC.

Vlastnosti:

- *Driver* - je jméno použitého ODBC ovladače.
- *Name* - vlastnost obsahuje cestu k souboru databáze Access.
- *Uid* - vlastnost obsahuje identifikaci uživatele, pokud je pro připojení požadována.
- *Psw* - vlastnost obsahuje přihlašovací heslo uživatele, pokud je pro připojení k souboru Access požadováno.
- *Connection* - vlastnost uchovává odkaz na objekt třídy *OdbcConnection*.
- *Connected* - logická hodnota, informuje o tom, zda aplikace je právě připojena k souboru databáze Access.
- *Initialized* - logická hodnota, informuje o tom, zda aplikace má nastaveny všechny parametry připojení k souboru databáze Access.
- *ExceptionMessage* – vlastnost obsahuje textový řetězec chybové zprávy.



Obr. 21. Diagram tříd aplikace pro migraci dat inventarizace lesů

Metody:

- *Access* - konstruktor nového objektu třídy *Access*.
- *Initialize* a *InitializeConnection* - metody provádějí první inicializaci nastavení parametrů připojení nebo jejich změnu.
- *SetConnection* - metoda zobrazuje dialog pro výběr souboru databáze *Access*, který má být otevřen a připojen.
- *Open* - metoda otvírá připojení k souboru databáze *Access*, případně ošetřuje vzniklé výjimky.
- *Close* - metoda uzavírá připojení k souboru databáze *Access*.
- *CancelConnection* - metoda ruší aktuální nastavení připojení k souboru databáze *Access*.
- *ConnectionInfo* - metoda vrací textový řetězec s informacemi o aktuálním připojení souboru databáze *Access*. Informace jsou zobrazovány na stavovém řádku hlavního formuláře.
- *TestQuery* - metoda testuje zadaný SQL příkaz a vrací logickou hodnotu pravda, pokud je příkaz proveditelný.
- *QueryResult* - metoda využívá komponenty *OdbcDataAdapter* pro provedení zadaného SQL dotazu a naplnění objektu třídy *DataTable* výsledkem SQL dotazu. Výsledek SQL dotazu je v podobě datové tabulky vrácen jako návratová hodnota metody. Metoda ošetřuje případné výjimky při provedení SQL příkazu a zobrazuje chybová hlášení.
- *QueryColumns* - metoda vrací objekt třídy *DataTable*, který obsahuje seznam datových sloupců a informace o datových sloupcích, které jsou obsaženy ve výsledku zadaného SQL příkazu.
- *GetTableList* - metoda vrací seznam tabulek obsažených v souboru databáze *Access*.
- *GetColumnsList* - metoda vrací seznam datových sloupců zvolené tabulky databáze *Access*.

6.2.9 Třída Npg

Objekty třídy Npg slouží v aplikaci pro migraci dat, k připojení a práci s databázemi PostgreSQL. Třída používá pro připojení a práci s databází třídy rozhraní *NpgSql*.

Vlastnosti:

- *Server* - textový řetězec s IP adresou nebo jménem databázového serveru PostgreSQL.
- *Port* - textový řetězec obsahující číslo TCP portu serveru PostgreSQL.
- *Name* - vlastnost obsahuje jméno databáze.
- *Uid* - je textový řetězec obsahující jméno přihlašovací role.
- *Psw* - je textový řetězec obsahující heslo přihlašovací role.
- *Connection* - vlastnost obsahuje odkaz na objekt třídy *NpgsqlConnection*.
- *Connected* - logická hodnota identifikující zda aplikace je připojena k databázovému serveru PostgreSQL.
- *Initialized* - logická hodnota informuje o tom, zda aplikace má nastaveny všechny parametry připojení k databázovému serveru PostgreSQL.

Metody:

- *Npg* - konstruktor objektu třídy *Npg*.
- *Initialize* a *InitializeConnection* - metody provádějí první inicializaci nastavení parametrů připojení nebo jejich změnu.
- *SetConnection* - metoda zobrazuje formulář pro nastavení parametrů připojení k databázovému serveru PostgreSQL.
- *Open* - metoda otvírá připojení k definované databázi na serveru PostgreSQL a případně ošetřuje vzniklé výjimky.
- *Close* - metoda uzavírá otevřené připojení k databázi na serveru PostgreSQL.
- *CancelConnection* - metoda ruší aktuální nastavení připojení k databázi na serveru PostgreSQL.
- *ConnectionInfo* - metoda vrací textový řetězec s informacemi o aktuálním připojení k databázi na serveru PostgreSQL. Informace jsou zobrazovány na stavovém řádku hlavního formuláře.
- *ExecuteNonQuery* - metoda nad definovaným připojením k databázi PostgreSQL vytvoří nový objekt třídy *NpgsqlCommand*, nastaví text SQL příkazu a zavolá metodu *ExecuteNonQuery*. Metoda zobrazí chybové hlášení v případě vzniku

výjimky. Metoda je používána k spuštění příkazů *INSERT* při vkládání nových dat do cílové databáze.

- *QueryResult* - metoda využívá komponenty *NpgsqlDataAdapter* pro provedení zadaného SQL dotazu a naplnění objektu třídy *DataTable* výsledkem SQL dotazu. Výsledek SQL dotazu je v podobě datové tabulky vrácen jako návratová hodnota metody. Metoda ošetřuje případné výjimky při provedení SQL příkazu a zobrazuje chybová hlášení.
- *QueryColumns* - metoda vrací objekt třídy *DataTable*, který obsahuje seznam datových sloupců a informace o datových sloupcích, které jsou obsaženy ve výsledku zadaného SQL příkazu.
- *LoRead* - metoda vrací objekt třídy *MemoryStream*, který obsahuje binární data z *large object* uložených v databázi systému PostgreSQL. Metoda slouží k načítání dat v databázi uložených fotografií a komprimovaných souborů projektů měření. Pro čtení binárních dat využívá metody objektu třídy *LargeObjectManager*.
- *LoSave* - metoda načítá *large object* identifikovaný číslem *oid* z databáze a načtený objekt ukládá do souboru určeného umístěním na disku.
- *LoUnZip* - metoda načítá *large object* identifikovaný číslem *oid* z databáze. Pokud načtený objekt obsahuje komprimovaný zip soubor projektu měřených dat. Soubor rozbálí do určeného umístění na disku.
- *LoImport* - metoda vytvoří nový *large object* a data zvoleného souboru importuje do vytvořeného *large object* v databázi.
- *LoSetOwner* - metoda nastavuje nového vlastníka existujícímu *large object* v databázi, který je určen identifikačním číslem *oid*.
- *LoDelete* - metoda odstraňuje zvolený *large object* určený identifikačním číslem *oid* z databáze.
- *GetSchemaNames* - metoda vrací seznam schémat obsažených ve zvolené databázi
- *GetUserGroups* - metoda vrací seznam skupinových rolí databázového serveru PostgreSQL.
- *GetTableList* - metoda vrací seznam datových tabulek ve zvoleném schématu.
- *GetColumnList* - metoda vrací seznam datových sloupců zvolené tabulky.

6.2.10 Třída `ApplicationSettings`

Třída *ApplicationSettings* je statickou třídou, od které nejsou vytvářeny žádné instance třídy. Třída obsahuje skupinu veřejných globálních proměnných, které jsou používány v rámci celé aplikace. Hodnoty proměnných jsou inicializovány při spuštění aplikace načtením z konfiguračního souboru pomocí metody *Load*. Při ukončení práce s aplikací jsou aktuální hodnoty globálních proměnných uloženy zpět do konfiguračního souboru metodou *Save*, aby mohly být použity při následujícím spuštění aplikace.

ZÁVĚR

Diplomová práce řeší úkol migrace provozních dat měření inventarizace lesů. Import provozních dat z projektů měření inventarizace lesů do cílové databáze probíhá většinou jednorázově po dokončení období měření. Data před vlastní migrací byla již zkontrolována, opravena a uložena ve formě jednotlivých souborů projektů v dočasném datovém úložišti. V průběhu importu dat do cílové databáze také není třeba řešit složité transformace dat. Struktura tabulek ve zdrojových projektech většinou odpovídá struktuře tabulek v cílové databázi. Problém úkolu spíše spočíval ve velkém množství měřených veličin a ve velkém množství souborů se zdrojovými daty, které bylo nutné integrovat do tabulek cílové databáze.

V rámci diplomové práce byla vytvořena jednoduchá aplikace, která umožňuje uživateli nahrávat měřená data projektů inventarizace lesů do cílové databáze. V období měření docházelo ke změnám ve struktuře měřených dat a byly vytvářeny upravené verze měřených projektů. Aplikace byla proto navržena tak, aby umožňovala uživatelům vytvářet své konfigurační soubory pro jednotlivé verze projektů. Výhodou specializované aplikace je snadnost jejího použití. Práce s aplikací v podstatě spočívá v otevření zdrojového projektu, určení cílové databáze a výběru vhodného konfiguračního souboru podle aktuální verze zdrojového projektu a spuštění přenosu dat vybraných inventarizačních ploch. Aplikace byla také rozšířena o možnost vytváření nových konfiguračních souborů pro další verze budoucích projektů měřených dat. Další výhodou je možnost tuto aplikaci dále upravovat podle aktuálně vznikajících potřeb uživatelů. Nevýhodou aplikace oproti obecným nástrojům pro migrace dat je její specifické zaměření pouze na konkrétní úkol migrace dat projektu inventarizace lesů. Nevýhodou je také poměrně velká počáteční pracnost s tvorbou programového kódu a jeho ladění a následná údržba.

Součástí diplomové práce bylo také prostudování možností řešit úkol migrace měřených dat některým z existujících softwarových nástrojů. Vybrány byly čtyři aplikace pro migraci dat, Clover ETL, Apatar, Scriptella a SQL Workbench/J. Do výběru byly záměrně zahrnuty dvě aplikace Clover ETL a Apatar, které mají grafické prostředí pro návrh procesu nahrání, transformace a uložení dat a dvě aplikace Scriptella a SQL Workbench/J, ve kterých je ETL proces popsán xml skriptem nebo pomocí rozšíření jazyka SQL.

Úkol migrace měřených dat je prakticky možné řešit kterýmkoliv z uvedených nástrojů. Výhodou aplikací s možností vytváření transformačních grafů v grafickém vývojovém

prostředí je rychlost vytvoření funkčního návrhu bez nutnosti tvorby kódu. V případě aplikace Clover je kód ve formě xml souboru generován z grafického návrhu transformačního grafu. Velkou výhodou aplikace Clover je také režim ladění jednotlivých částí transformačního grafu a možnost tvorby vlastních komponent. Použití programu Clover je však omezeno skutečností, že se jedná o aplikaci komerční.

SEZNAM POUŽITÉ LITERATURY

- [1] LACKO Luboslav. *Business Intelligence v SQL Serveru 2008: reportovací, analytické a další datové služby*. Vyd. 1. Brno: Computer Press, 456 s. ISBN 978-80-251-2887-9.
- [2] KIMBALL, Ralph. *The data warehouse lifecycle toolkit*. 2nd ed. Indianapolis, IN: Wiley Pub., c2008, xxxiv, 636 p. ISBN 04-701-4977-9.
- [3] RAINARDI, Vincent. *Building a data warehouse with examples in SQL Server*. Berkley, CA: Apress; Distributed to the book trade worldwide by Springer-Verlag New York, c2008. ISBN 15-905-9931-4
- [4] SCHILLER Martin. *Co se skrývá pod zkratkou ETL? Jak zpracovat informace uložené v různých podnikových systémech*. [online]. [cit. 2013-2-12]. Dostupné z: <http://www.systemonline.cz/clanky/co-se-skryva-pod-zkratkou-etl.htm>
- [5] LABERGE, Robert. *Datové sklady: agilní metody a business intelligence*. 1. vyd. Brno: Computer Press, 350 s ISBN 978-80-251-3729-1.
- [6] VAVRUŠKA Jindřich. *ETL a kvalita dat*. [online]. [cit. 2013-2-12]. Dostupné z: <http://www.systemonline.cz/clanky/etl-a-kvalita-dat.htm>
- [7] BALLARD Chuck, HERREMAN Dirk, SCHAU Don, BELL Rhonda. *Data Modeling Techniques for Data Warehousing*. [online] [cit. 2014-4-3]. Dostupné z: <https://www.redbooks.ibm.com/redbooks.nsf/RedbookAbstracts/sg242238.html>
- [8] VASSILIADIS Panos, SIMITSIS Alkis. *Extraction, Transformation, and Loading* [online]. [cit. 2013-2-12]. Dostupné z: <http://www.dblab.ntua.gr/~asimi/publications/VaSi09-etl-ency.pdf>
- [9] WALLER Tomas, STYS Miroslav et al. *Clover ETL Designer: User's Guide* [online]. [cit. 2014-3-27]. Dostupné z: <http://www.cloveretl.com/resources>
- [10] *Aptar Quick Start Guide* [online]. [cit. 2014-3-27]. Dostupné z: http://www.apatar.com/pdf/Aptar_Quick_Start_Guide.pdf

- [11] *Scriptella ETL Reference Documentation* [online] [cit. 2014-3-29]. Dostupné z: <http://scriptella.javaforge.com/reference/index.html>
- [12] *SQL Workbench/J User's Manual* [online] [cit. 2014-3-31]. Dostupné z: <http://www.sql-workbench.net/manual/workbench-manual.html>
- [13] MOMJIAN, Bruce a Richard STONES. *PostgreSQL: introduction and concepts*. 2nd ed. Boston, MA: Addison-Wesley, 2001, xxiv, 637 p. ISBN 02-017-0331-9.
- [14] *PostgreSQL 9.2.8 Documentation* [online] [cit. 2014-4-5] Dostupné z: <http://www.postgresql.org/files/documentation/pdf/9.2/postgresql-9.2-US.pdf>
- [15] ARGAWAL, Vidya Vrat a James HUDDLESTON. *Databáze v C# 2008: průvodce programátora*. Vyd. 1. Brno: Computer Press, 2009, 424 s. ISBN 978-80-251-2309-6.
- [16] MATTHEW, Neil a Richard STONES. *Beginning databases with PostgreSQL: from novice to professional*. 2nd ed. Berkeley, CA: Apress, c2005. ISBN 15-905-9478-9.

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

SQL	Structured Query Language. Programovací jazyk navržený pro práci s daty v relačních databázových systémech.
OLTP	Online Transaction Processing. Technologie uložení dat v databázích, optimalizovaná pro rychlé vkládání nových záznamů a snadnou modifikaci existujících.
OLAP	Online Analytical Processing. Technologie uložení dat a analytické služby, které slouží pro analýzu velkého množství dat.
ETL	Zkratka extract, transform a load, označuje postup získávání dat z provozních systémů, jejich transformaci, čištění a uložení v datových úložištích cílových systémů.
J2EE	Java Platform Enterprise Edition je platforma pro vývoj aplikací založená na jazyce Java.
ODBC	Open Database Connectivity. Softwarové rozhraní pro přístup k databázovým systémům, vyvinuté společností Microsoft.
JDBC	Java Database Connectivity. Softwarové rozhraní pro přístup k databázovým systémům, založené na programovacím jazyce Java.
URL	Uniform Resource Locator. Textový řetězec sloužící k jednoznačné identifikaci zdroje.
CTL	Clover Transformation Language. Jazyk pro definici transformací a zpracování dat v rámci komponent transformačního grafu aplikace Clover.
XML	Extensible Markup Language. Obecný značkovací jazyk.
DTD	Document type definition. Jazyk pro definici struktury XML dokumentu.
S-JTSK	Jednotná trigonometrická síť katastrální. Pravoúhlá souřadnicová síť pro mapy území České Republiky.

SEZNAM OBRÁZKŮ

<i>Obr. 1. Clover ETL – formulář definice metadat</i>	<i>25</i>
<i>Obr. 2. Clover ETL – formulář definice připojení.....</i>	<i>26</i>
<i>Obr. 3. Clover ETL - transformační graf pro import dat</i>	<i>27</i>
<i>Obr. 4. Clover ETL - transformační graf pro export dat.....</i>	<i>29</i>
<i>Obr. 5. Apatar - datová mapa</i>	<i>31</i>
<i>Obr. 6. Apatar - definice mapování sloupců.....</i>	<i>32</i>
<i>Obr. 7. SQL Workbench/J - dialog Manage drivers</i>	<i>36</i>
<i>Obr. 8. SQL Workbench/J - dialog Select Connection Profile</i>	<i>37</i>
<i>Obr. 9. Diagram tabulek databáze</i>	<i>56</i>
<i>Obr. 10. Hlavní formulář aplikace pro migraci dat inventarizace lesů</i>	<i>58</i>
<i>Obr. 11. Formulář připojení k databázi PostgreSQL.....</i>	<i>60</i>
<i>Obr. 12. Dialog nahrazení starých dat inventarizační plochy</i>	<i>62</i>
<i>Obr. 13. Formulář průběh importu dat</i>	<i>62</i>
<i>Obr. 14. Formulář výsledek importu dat</i>	<i>63</i>
<i>Obr. 15. Formulář definice importu dat tabulek</i>	<i>65</i>
<i>Obr. 16. Formulář nový konfigurační soubor</i>	<i>66</i>
<i>Obr. 17. Formulář tabulky cílové databáze.....</i>	<i>67</i>
<i>Obr. 18. Formulář mapování sloupců</i>	<i>68</i>
<i>Obr. 19. Formulář fotografie inventarizačních ploch</i>	<i>69</i>
<i>Obr. 20. Diagram tříd – seznamy inventarizačních ploch.....</i>	<i>72</i>
<i>Obr. 21. Diagram tříd aplikace pro migraci dat inventarizace lesů</i>	<i>81</i>