FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

ZADÁNÍ DIPLOMOVÉ PRÁCE

Název:	Analýza datových toků v Excelu
Student:	Bc. Daniel Míček
Vedoucí:	Ing. Michal Valenta, Ph.D.
Studijní program:	Informatika
Studijní obor:	Webové a softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2019/20

Pokyny pro vypracování

1. Nastudujte možnosti jak Microsoft Excel získává data ze zdrojových databází, jak je transformuje a jak prezentuje výsledky.
2. Seznamte se s projektem Manta a způsobem, jak reprezentuje datové toky.
3. Navrhněte způsob jak detekovat datové struktury v Excelu a jak je následně reprezentovat v datovém modelu Manty.
4. Navrhněte způsob jak mezi nalezenými strukturami v Excelu detekovat datové toky.
5. Realizujte prototyp nástroje, který ze sady souborů ve formátu Excel extrahuje datové toky do systému Manta.

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
děkan

V Praze dne 30. ledna 2019

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Diplomová práce

Analýza datových toků v Excelu

Bc. Daniel Míček

Katedra softwarového inženýrství
Vedoucí práce: Ing. Michal Valenta, Ph.D.

7. května 2019

Poděkování

Na tomto místě bych rád poděkoval Ing. Michalu Valentovi, Ph.D. za vedení diplomové práce. Dále bych chtěl poděkovat Lukáši Hermannovi a Jaroslavu Kotrčovi ze společnosti Manta za cenné rady a připomínky.

Prohlášení

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

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů (dále jen „autorský zákon"), především § 35 a § 60 autorského zákona upravující školní dílo.

V případě počítačových programů, jež jsou součástí mojí práce či její přílohou, a veškeré související dokumentace k počítačovým programům (dále jen „software"), uděluji v souladu s ust. § 2373 zákona 89/2012 Sb., občanský zákoník, ve znění pozdějších předpisů, nevýhradní a neodvolatelné oprávnění (licenci) k užití software, a to všem osobám, které si přejí software užít. Tyto osoby jsou oprávněny software užít jakýmkoli způsobem a za jakýmkoli účelem v neomezeném rozsahu (včetně užití k výdělečným účelům), vč. možnosti software upravit či měnit, spojit jej s jiným dílem a/nebo zařadit jej do díla souborného. Toto oprávnění je časově, teritoriálně i množstevně neomezené a uděluji jej bezúplatně.

V Praze dne 7. května 2019 .....

Odkaz na tuto práci

Míček, Daniel. *Analýza datových toků v Excelu*. Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2019.

Abstrakt

Cílem této práce je analyzovat způsob ukládání dat v souborech Microsoft Excel a navrhnout datový model vhodně popisující strukturu objektů, které se v sešitech Excelu mohou objevit. Dále je nutné navrhnout a implementovat rozšíření produktu Manta Flow zpracovávající Excelovské soubory. Výsledkem zpracování je graf, kde jednotlivé uzly reprezentují objekty Excelovského sešitu a hrany reprezentují datové toky mezi nimi.

Klíčová slova Analýza datových toků, datový tok, Microsoft Excel, Manta Flow.

Abstract

The aim of this thesis is to analyze the way data is stored in Microsoft Excel files and to design a data model that appropriately describes the structure of the objects that can appear in Excel worksheets. Furthermore it is necessary to design and implement extension of the product Manta Flow that processes Excel files. The result of the processing is a graph, where each node represents an object of an Excel sheet and edges represent data flows of those nodes.

Keywords Dataflow analysis, dataflow, Microsoft Excel, Manta Flow.

Obsah

Seznam obrázků

Úvod

S narůstajícím objemem dat v BI[1] prostředí narůstá i potřeba se v těchto datech orientovat. Problematika zabývající se původem a pohybem dat v takovém systému se nazývá Data Lineage. Data v BI systému mohou pocházet z různých technologií a během svého životního cyklu v systému je možné s nimi pracovat v úplně jiných technologiích. Výstupní podobou dat z BI prostředí bývá sumarizační report.

Existuje mnoho reportovacích nástrojů, jako je například IBM Cognos či Microsoft SSRS, pomocí kterých lze přijímat externí data ze systému a vypracovat nad nimi výstupní report. I přes jejich existenci se však v business prostředí k tomuto účelu využívá také tabulkový nástroj Microsoft Excel, především díky jeho jednoduchosti a dostupnosti.

Microsoft Excel je tabulkový procesor, jehož první verze vyšla již v roce 1985. Díky jeho popularitě je v současnosti Excel považován jako synonymum tabulkového procesoru. Excel umožňuje uživateli snadno a rychle vytvořit tabulku či graf. Nástroj Microsoft Excel bude představen v první kapitole.

Ve druhé kapitole bude představena společnost Manta a její nástroj Manta Flow, jenž umožňuje uživateli analyzovat své BI prostředí a vizualizovat datové toky mezi jeho objekty. Tato práce si dává za cíl navrhnout a implementovat prototyp rozšíření tohoto nástroje, který dokáže analyzovat soubory Microsoft Excel a vizualizovat toky mezi jeho objekty.

V rámci další kapitoly budou ve stručnosti popsány cíle implementace prototypu tohoto rozšíření. Mimo jiné budou představeny technologie použité k jeho realizaci.

Čtvrtá kapitola se zaměří na analýzu konkrétní struktury souborů, ze kterých se skládá sešit Excelu. Zmíněny budou jednotlivé objekty, které jsou v těchto souborech definovány, jejich vlastnosti a vztahy. Mimo jiné za účelem hledání datových toků je důležitá analýza definic externích dotazů, ze kterých čerpá Excel data.

[1] Business Intelligence

V páté kapitole bude dán prostor návrhu datového modelu a gramatik parseru jazyků, které se v Excelu nachází.

Poslední kapitola patří samotné implementaci prototypu. V této kapitole bude stručně představena knihovna Apache POI umožňující přístup k určitým objektům Excelovského sešitu z programovacího jazyka Java. Dále bude stručně popsán princip dvou fází analýzy Excelovského souboru, kterými jsou resolving a generování dataflow grafu. Z těchto fází pak budou představeny konkrétní zajímavé části řešení implementace. Samozřejmě nebude opomenuto testování.

Kapitola 1

Excel

Excel je tabulkový procesor a součást nástrojové sady Microsoft Office. Jeho první verze vyšla v září 1985 pro Apple Macintosh. Od verze Microsoft Office 2007 používá Excel a ostatní kancelářské programy balíku Microsoft Office formát OOXML[2] pro ukládání souborů (s příponou `xlsx`), který nahradil proprietární binární soubory formátu OLE2[3] (s příponou `xls`).

Předchůdcem Excelu byl program Multiplan, který byl vyvinut firmou Microsoft v roce 1982 jako soupeř programu VisiCalc, který byl dostupný na počítače Apple II od roku 1979 a později IBM PC. VisiCalc zavedl styl referencí `A1`. Multiplan byl dostupný pro CP/M počítače a později byl portován na další platformy, mimo jiné na MS DOS[4] a Apple II. Multiplan oproti ostatním tabulkovým programům té doby používal reference stylu `R1C1`. V roce 1983 byl vydán program Lotus 1-2-3 firmou Lotus Software (později součást IBM), který překonal prodeje programu VisiCalc a stal se velkou součástí úspěchu IBM PC. Lotus 1-2-3 kromě tabulkových výpočtů obsahoval i databázovou funkcionalitu a grafy, z tohoto důvodu obsahoval v názvu "1-2-3".[3]

První verze Excelu, jak už bylo zmíněno, vyšla v roce 1985 pro Apple Macintosh. O dva roky později, v listopadu 1987, vyšel Excel 2.0 pro Microsoft Windows. V roce 1990 přišla nová verze Excel 2.1d, která byla kompatibilní s Windows 3.0. Verze Excel 3.0 nově obsahovala nástroj Řešitel, podporu rozšíření a podporu technologie Object Linking and Embedding (zkráceně OLE). S verzí 5.0 byla uvedena podpora jazyka VBA[5] pro uživatelem definované funkce. Další verze, konkrétně Excel 95, Excel 97, Excel 2000, Excel 2002 a Excel 2003, přinesly jen mírná zlepšení. [3] Verze Excel 2007 s sebou přinesla nový souborový formát OOXML, posunula limity velikosti souborů a vylepšila využití více procesorových jader na výpočty.

Excel organizuje data do mřížky buněk, kde se jednotlivé buňky inde-

[2] Office Open XML
[3] uváděno někdy i jako Object Linking and Embedding (OLE) Compound File (CF)
[4] Microsoft Disk Operating System
[5] Visual Basic for Applications

xují řádkově jako čísla a sloupcově jako velká písmena anglické abecedy. Tato metoda adresování je nazývána jako reference stylu `A1`. Excel však disponuje i druhým způsobem psaní referencí, který se nazývá styl `R1C1`. Reference stylu `R1C1` je pozůstatek po tabulkovém programu Multiplan. Reference psané v tomto stylu zapisují index řádků i sloupců jako číslo, před kterým se nachází `R` nebo `C` pro řádek a sloupec respektivně. Styl referencí lze přepnout v nastavení aplikace. K manipulaci s daty a aritmetickým operacím slouží vlastní jazyk Excelovských vzorců obsahující různorodé vestavěné funkce řazené do čtrnácti kategorií. Vlastní funkce je možné definovat pomocí jazyka VBA.

Data lze do Excelu vložit ručním vepsáním nebo je možné importovat data ze souboru či externího zdroje. Pro usnadnění importování dat z externích datových zdrojů nabízí Excel nástroje Microsoft Query, jehož funkci v novějších verzích převzal nástroj Power Query[6].

1.1 Office Open XML

Office Open XML (zkráceně OOXML) je specifikace souborového formátu pro dokumenty kancelářské sady Microsoft Office od verze 2007. Specifikace byla standardizována organizací ECMA[7] jako ECMA-376. Další standardizací tohoto formátu je ISO/IEC 29500.[4]

Standard ECMA-376 obsahuje specifikace pro tři hlavní typy dokumentů. WordprocessingML pro textové dokumenty, SpreadsheetML pro tabulkové dokumenty a PresentationML pro prezentace. Za zmínku stojí ještě specifikace DrawingML, která je využívaná pro vykreslování tvarů a grafů.

Pro všechny druhy souborů specifikace OOXML jsou společné soubory ve složce `docProps`. V této složce se nachází dva soubory. Soubor `core.xml`, který obsahuje informace o datu vytvoření, datu poslední modifikace a o osobách, které tyto akce provedly. Druhým souborem je `app.xml` obsahující informace o souboru, které jsou specifické pro jeho typ. V případě Excelu obsahuje počet listů s jejich názvy, verzi Excelu, ve které byl soubor vytvořen či údaje o aktuálnosti externích dat.

Uložení samotného dokumentu na disk je pak specifikováno pomocí OPC (Open Packaging Conventions). OPC využívá běžného ZIP formátu pro vytvoření balíku XML souborů. Jednotlivé dílčí XML soubory mohou obsahovat různá data a pro určení jejich typů je místo přípon jednotlivých souborů použit soubor `[Content_Types].xml`.

Pro propojení jednotlivých souborů uvnitř OOXML souboru jsou použity vztahy, které jsou definované mimo samotný soubor. Jejich specifikace se nachází v souboru se stejným názvem a příponou `.rels` ve složce `_rels` v místě původního souboru, který referenci využívá. Samotné záznamy vztahů obsahují identifikátor vzathu, jeho typ a cíl.

[6] Také známý jako Get & Transform od verze Microsoft Excel 2016

[7] European Computer Manufacturers Association

Mimo standardního formátu sešitu specifikace OOXML s koncovkou `xlsx` existují i další varianty. Formát s koncovkou `xlsm` označuje sešit s podporou maker. Tento druh sešitů si zachovává stejnou strukturu xml souborů jen s tím rozdílem, že je mezi nimi přítomný soubor `vbaProject.bin` obsahující veškeré skripty, které jsou v sešitu definované. Tyto skripty jsou psány v jazyce Visual Basic for Applications.

Druhým ze speciálních formátů sešitu je binární sešit s koncovkou `xlsb`. Tento druh sešitu zachovává XML podobu OOXML pro složku `docProps`, avšak veškerá data specifická pro Excel jsou ukládána v podobě binárního souboru zachovávajícího stejnou strukturu jako klasický sešit. Uložením sešitů v tomto formátu lze dosáhnout jejich rychlejšího ukládání a otevírání, obzvláště u těch obsahujících velké množství dat.

Kapitola 2

Manta tools

Manta Tools, zkráceně Manta, je česká startupová společnost vyčleněná z firmy Profinit, kde původně v letech 2008 a 2009 vznikla jako interní nástroj. Manta v roce 2013 získala grant od Technologické Agentury ČR na další rozvoj. Koncem roku 2014 Manta získala první místo ve výběrovém řízení Czech ICT Incubator pořádaným Czech ICT Alliance, čímž získala příležitost akcelerovat svojí expanzi 3-měsíčním pobytem v Silicon Valley. [5]

Nejčastějšími zákazníky produktů Manty jsou zahraniční firmy, mezi které patří například Paypal, OBI, Vodafone či Comcast. Současnými produkty firmy Manta jsou Manta Flow a Manta Checker. [6]

2.1 Manta Flow

Nástroj Manta Flow se zabývá vizualizováním data lineage napříč komplexním systémem pomocí analýzy kódu a detailní vizualizace toků dat mezi jednotlivými objekty v Business Intelligence prostředí. Využitím této vizualizace mohou být například dopadové analýzy, optimalizace datových skladů či migrace dat.

Výsledkem analýzy datových toků je orientovaný graf, který je tvořen uzly a orientovanými hranami. Uzly v tomto případě reprezentují hierarchickou strukturu objektů některé z analyzovaných technologií. Příklady těchto uzlů mohou být například sloupec, tabulka či databáze. Uzly jsou uvnitř grafu Manta Flow jednoznačně identifikovány pomocí kombinace názvu, rodičovského uzlu, typu a technologie. Jakékoli další potřebné vlastnosti lze uzlu volitelně přiřadit jako jeho atributy.

Orientovanými hranami v grafu jsou reprezentovány jednotlivé toky dat. Tyto hrany spojují dva uzly, což znamená, že zdrojový uzel nějakým způsobem ovlivňuje hodnotu uzlu cílového. Datové toky jsou dvojího typu.

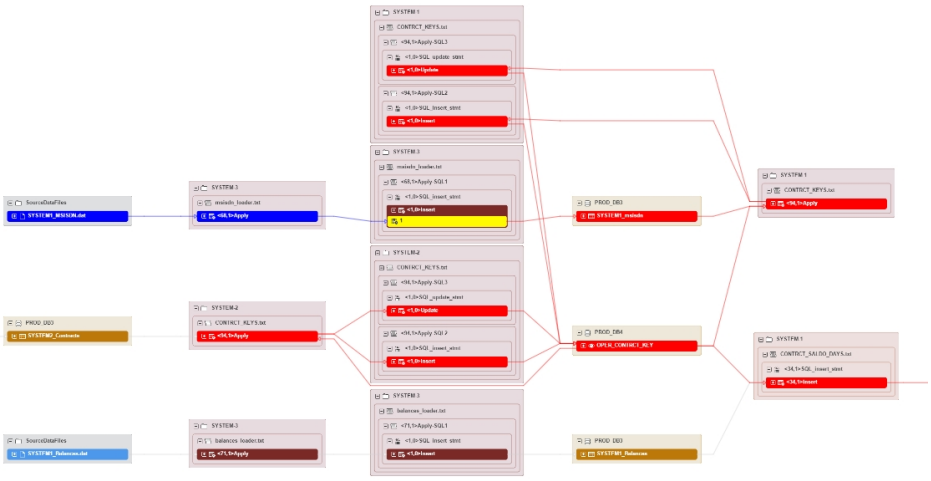
Přímé datové toky, které označují, že zdrojový uzel se přímo podílí na vzniku hodnoty cílového uzlu. Příkladem přímých datových toků je sloupec tabulky, který vznikne součtem hodnot ze dvou jiných sloupců.

Nepřímé datové toky jsou někdy označované jako filtrující datové toky. Tyto toky ovlivňují obsah cílového uzlu, aniž by se na něm přímo podílely. Jako příklad nepřímého toku lze uvést obsah klauzule `WHERE` v SQL dotazu, která omezuje množinu hodnot, ze kterých vznikla hodnota cílového uzlu.

Podporovanými technologiemi jsou v současnosti tyto:

- Teradata Database
- Informatica PowerCenter
- Oracle Database
- SAP ASE (Sybase)
- Hive
- IBM Netezza
- Microsoft SQL Server Database
- Microsoft SSIS, SSAS, SSRS
- Microsoft APS
- Microsoft Azure SQL Database
- Microsoft Azure SQL DWH
- IBM DB2
- Impala
- PostgreSQL
- Oracle Data Integrator
- Amazon Redshift
- ER/Studio
- Greenplum
- Sqoop
- Pig
- Talend
- Java

[7]



Obrázek 2.1: Snímek obrazovky z vizualizace Manta Flow.[1]

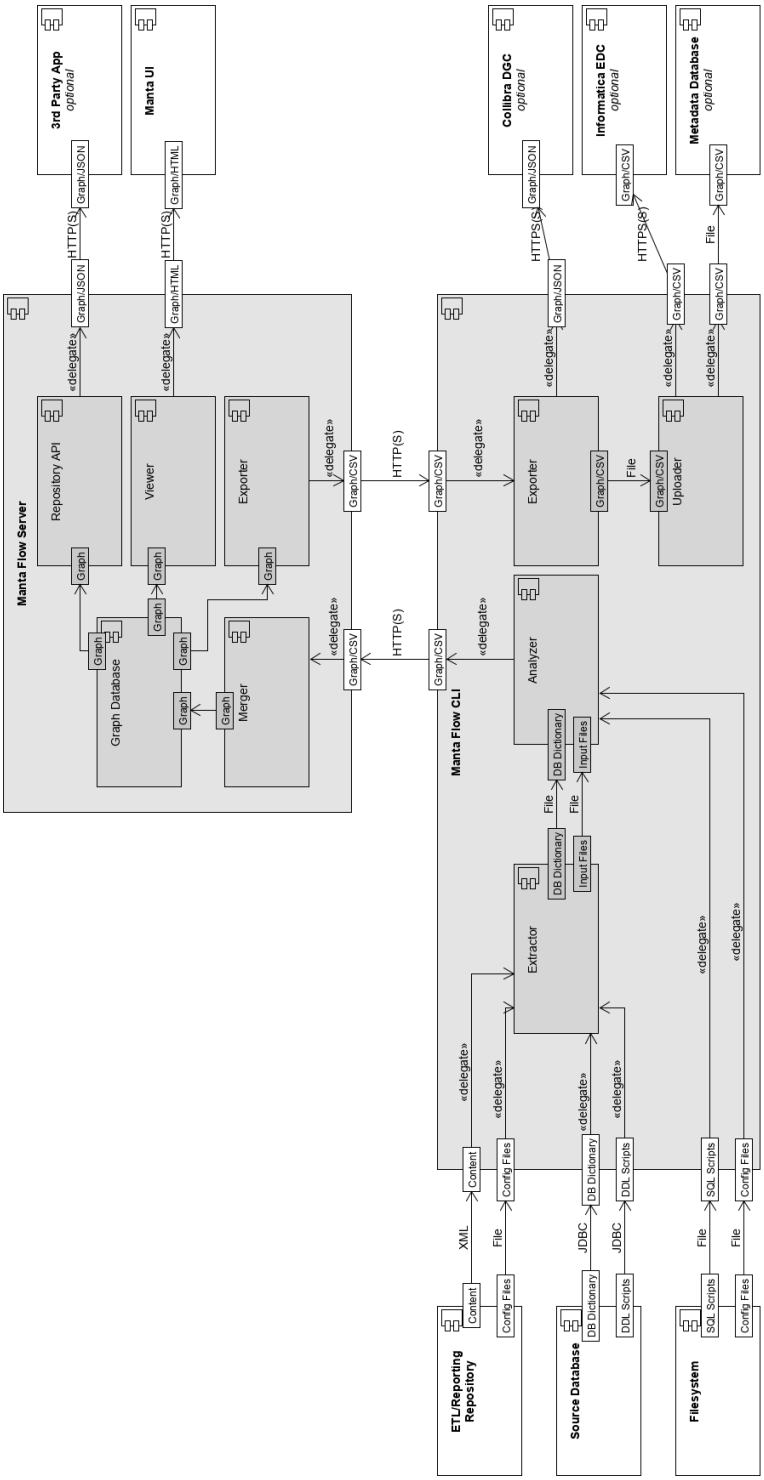
2.1.1 Architektura

Manta Flow je aplikace architektury client-server. Kompletní podoba architektury je vyobrazena na diagramu 2.2. Klientská část aplikace je aplikace příkazové řádky psaná v jazyce Java, která komunikuje s Manta Flow serverem pomocí TCP/IP protokolu.

Klientskou část tvoří komponenta **Manta Flow CLI**, jejímž učelem je extrakce databázových slovníků a skriptů z databáze uživatele, které jsou následně analyzovány. Výsledek analýzy v podobě uzlů a hran grafové databáze je nahrán na server.

Komponenta **Manta Flow Server** tvoří serverovou část aplikace, která uchovává metadata získané klientskou částí ve svém úložišti. Tyto metadata dále zpřístupňuje vizualizaci nebo aplikacím třetí strany pomocí veřejného API.

Kromě části Manta Flow Server se na serverovové části nachází i komponenta **Manta Flow Service Utility**, která má na starost konfiguraci a obnovování uživatelského rozhraní. [2]



Obrázek 2.2: Diagram architektury Manta Flow. [2]

Kapitola 3

Cíle implementace

Cílem implementace je vytvořit rozšíření produktu Manta Flow analyzující datové toky v souborech Microsoft Excel. Pro reprezentování objektů je nutno navrhnout hierarchický datový model odpovídající objektům, které se vyskytují v sešitech Excelu. Dále je nutné implementovat resolver, který z konkrétního analyzovaného Excelovského sešitu vytvoří konkrétní instance objektů datového modelu s odpovídajícími vlastnostmi. Výsledné objekty konkrétních instancí objektů datového modelu budou poté analyzovány pro vzájemné odkazy pomocí Excelovských vzorců a pro informace o původu dat z externích zdrojů. Samotné objekty a vztahy mezi nimi jsou poté v podobě grafu uloženy do grafové databáze.

Pro generování vizualizace datových toků uvnitř Excelovského sešitu bude nutné sestavit gramatiku parsovacího nástroje ANTLR[8], která vytvoří abstraktní syntaktický strom z Excelovských vzorců. Tímto parserem budou analyzovány jednotlivé vzorce a na základě referencí v nich budou referované objekty spojené datovým tokem se zdrojovým objektem.

Druhým jazykem, který je nutný analyzovat pro vytvoření datových toků, je jazyk definic dotazů Power Query M. Účelem parsování tohoto jazyka je zjištění druhu technologie a názvu referovaného externího objektu. S těmito informacemi lze dohledat cílový objekt v databázovém slovníku analyzovaných objektů a následně je spojit se zdrojovým objektem.

Kvůli jednoduchosti je Excel používán pro spoustu účelů, včetně vytváření reportů v business prostředí. Reporty v Excelu jsou důvodem zadání této práce, a proto bude více kladen důraz na nástroje využívané k jejich tvorbě.

3.1 Použité technologie

V této sekci budou představeny jednotlivé nástroje, pomocí kterých bude vytvořeno rozšíření produktu Manta Flow o technologii Excelu.

[8] ANother Tool for Language Recognition

3.1.1 Java

Java patří mezi nejpoužívanější programovací jazyky na světě. Java je objektově orientovaný a multiplatformní jazyk, pomocí něhož bude implementován modul Connector a Dataflow Generator pro Excelovské soubory. [8]

3.1.2 ANTLR

ANTLR je nástrojem generující parser z gramatiky. Gramatika obsahuje dvě části, které mohou být zapsány buď zvlášť v oddělených souborech nebo v jednom souboru komplexní gramatiky. První částí je gramatika lexeru, která definuje jednotlivé tokeny, které se v parsovaném jazyce nachází. S těmito tokeny je pak dále pracováno v druhé části, kterou je gramatika parseru. Pomocí gramatiky parseru se zapisují syntaktická pravidla parsovaného jazyku. [9]

3.1.3 Apache Maven

Maven je nástroj využívaný na sestavování projektu z více modulů, pomocí něhož je také možné spravovat závislosti jednotlivých použitých modulů nebo knihoven. Maven je vyvíjen společností Apache Software Foundation (ASF) jako open-source software. [10]

3.1.4 Spring

Framework Spring umožňuje implementovat programovací techniku vkládání závislosti, čímž je zvýšena modularita programu. Spring je hojně využívaný v aplikacích Java Enterprise Edition (Java EE). [11]

3.1.5 Subversion

Apache Subversion je open-source nástroj od ASF, který spravuje verzování zdrojových kódů. [12]

3.1.6 Knihovna Apache POI

Pro manipulaci se soubory Microsoft Excel v Javě je použita knihovna Apache POI. POI je open-source knihovna, která je stejně jako Subversion a Maven vyvíjena společností ASF.[13]

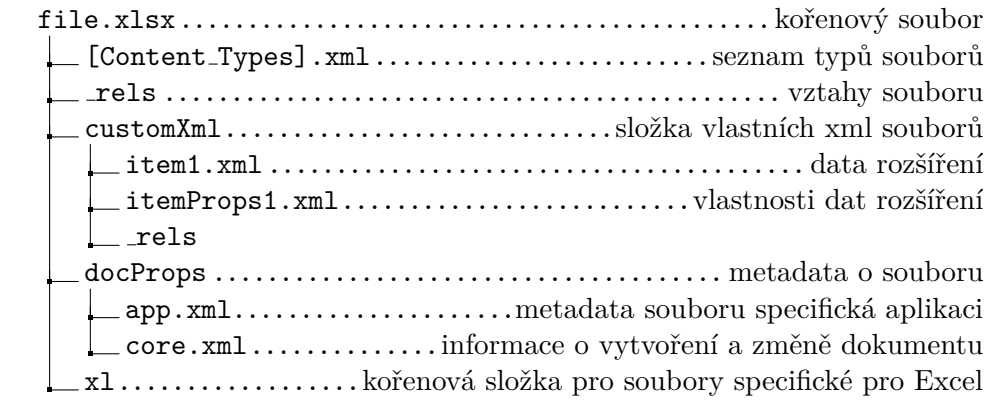
3.1.7 Knihovna dom4J

Open-source knihovna dom4j umožňuje parsovat XML soubory v Javě a přistupovat k elementům parsovaného XML souboru pomocí jazyka XPath. [14]

Kapitola 4

Analýza Excelovských souborů

V této kapitole bude představena struktura Excelovského souboru a jednotlivé elementy, které se v sešitu mohou vyskytnout. Kromě souborů, které jsou specifické pro Excel, jsou v balíčku souboru Excelovského sešitu soubory společné pro všechny OOXML soubory. Mezi tyto společné vlastnosti patří například informace o modifikacích.



Obrázek 4.1: Strom vnitřních souborů kořenového adresáře OOXML dokumentu

4.1 Excelovské objekty

Jednotlivé typy objektů, které se v Excelovském sešitu nacházejí, budou v této sekci představeny. Zmíněny budou pouze jejich vlastnosti, které jsou důležité pro generování vizualizace datových toků mezi dílčími objekty. Mimo obecných vlastností popisující určitý objekt, jako je jeho název, bude dbán důraz na jakékoliv indexy či reference, pomocí kterých se jednotlivé objekty odkazují na jiné. Vlastnosti týkající se vzhledu a stylování budou ignorovány.

```
xl ..... kořenová složka pro soubory specifické pro Excel
├── calcChain.xml ..... popisuje posloupnost vyhodnocování vzorců
├── connections.xml ..... obsahuje definované dotazy
├── sharedStrings.xml ..... tabulka sdílených textů
├── styles.xml ..... styly sešitu
├── workbook.xml ..... definiční soubor sešitu
├── vbaProject.bin ..... makra a uživatelem definované funkce
├── charts ..... definice grafů
│   └── chart1.xml
├── drawings ..... definice nákresů
│   └── drawing1.xml
├── externalLinks ..... cache externích xlsx souborů
│   └── externalLink1.xml
├── model ..... datový model rozšíření Power Pivot
│   └── item1.data
├── pivotCache ..... cache kontingenčních tabulek
│   ├── pivotCacheDefinition1.xml ..... definice cache
│   └── pivotCacheRecords1.xml ..... data cache
├── pivotTables ..... definice kontingenčních tabulek
│   └── pivotTable1.xml
├── queryTables ..... definice dotazových tabulek
│   └── queryTable1.xml
├── tables ..... definice Excelovských tabulek
│   └── table1.xml
├── slicers ..... definice průřezů
│   └── slicer1.xml
├── slicerCache ..... definice zdrojů průřezů
│   └── slicerCache1.xml
├── theme ..... styly drawingML
├── worksheets ..... definice a obsah listů
│   └── sheet1.xml
├── chartsheets ..... definice grafových listů
│   └── sheet1.xml
└── _rels ..... vztahy sešitu
```

Obrázek 4.2: Strom vnitřních souborů specifických pro Excel [15]

4.1.1 Sešit

Sešit je hlavní kořenový objekt Excelovského souboru spojující všechny Excelovské listy do jednoho celku. Vlastnosti sešitu jsou definovány v souboru `/xl/workbook.xml`. Jediné povinné atributy definičního souboru sešitu jsou záznamy o jeho listech. Kromě seznamu listů obsahuje odkazy na objekty

sdílené napříč všemi listy. Těmito objekty jsou:

- Definice cache kontingenčních tabulek
- Externí soubory referencovány z tohoto sešitu
- Definované názvy
- Databázové spojení
- Cache průřezů

[16]

4.1.2 Listy

Listy Excelovského souboru jsou centrální součástí sešitu a částí Excelu, se kterou uživatel nejvíce interaguje. Obsah a vlastnosti listů jsou uchovávány v jeho definičním souboru `/xl/worksheets/sheet1.xml`. Uvnitř tohoto souboru jsou také uchovávány informace o vyplněných buňkách, které obsahují nejen informace o jejich hodnotě, ale i o jejich stylu. Reference na každý list je uchovávana uvnitř souboru sešitu `/xl/workbook.xml`.

Samotná data vyplněných buněk jsou v souboru listu ukládána v elementu `sheetData` po řádcích. Jednotlivé řádky si uchovávají jednak svůj řádkový index, protože prázdné řádky definované být nemusí, jednak rozsah sloupcových indexů svých vyplněných buněk jako volitelný atribut.

Elementy buněk obsahují atributy své adresy a datového typu. V případě, že atribut typu buňky není uveden, předpokládá se, že typ buňky je číslo. Možnými datovými typy Excelu jsou:

- `boolean`
- `error`
- `inline string` (text s vlastními styly)
- `number` (využíváno pro čísla různých formátů nebo data)
- `shared string` (sdílený text v celém sešitu)
- `string`

Hodnota buňky se uchovává zvlášť v elementu potomka buňky. Speciálním případem jsou buňky typu `shared string`, kde hodnota buňky odpovídá indexu záznamu z tabulky sdílených textových řetězců, která je společná pro celý sešit.

Kromě elementu hodnoty buňky mohou obsahovat buňky i samostatný element vzorce, čímž je uchována poslední vypočtená hodnota i samotný vzorec,

kterým byla hodnota vypočtena. I přes přeložené názvy funkcí podle lokalizace distribuce Excelu se vnitřně vzorce ukládají s názvy funkcí v angličtině. Vzorce mohou být čtyř různých typů:

- `normal`
- `shared`
- `array`
- `dataTable`

Vzorce typu `dataTable` značí použití nástroje citlivostní analýzy tabulky dat. `Shared` vzorce jsou sdílené vícero buňkami, kde samotný vzorec je pouze v první buňce s určitou hodntou atributu `shared index` a referencí na rozsah buněk sdílející tento vzorec, a ostatní buňky téhož vzorce mají pouze tento index. Vzorce `array` označují použití vektorových nebo maticových výpočtů. Vzorce tohoto typu nejsou z pohledu analýzy datových toků důležité, neboť nebývají v reportech využívány.

Mimo standardních listů existují i listy bez mřížky buněk, které obsahují pouze jeden graf a jsou definované zvlášť od obyčejných listů v souborech `/xl/chartSheets/chartSheet1.xml`.[17]

4.1.3 Externí datová spojení

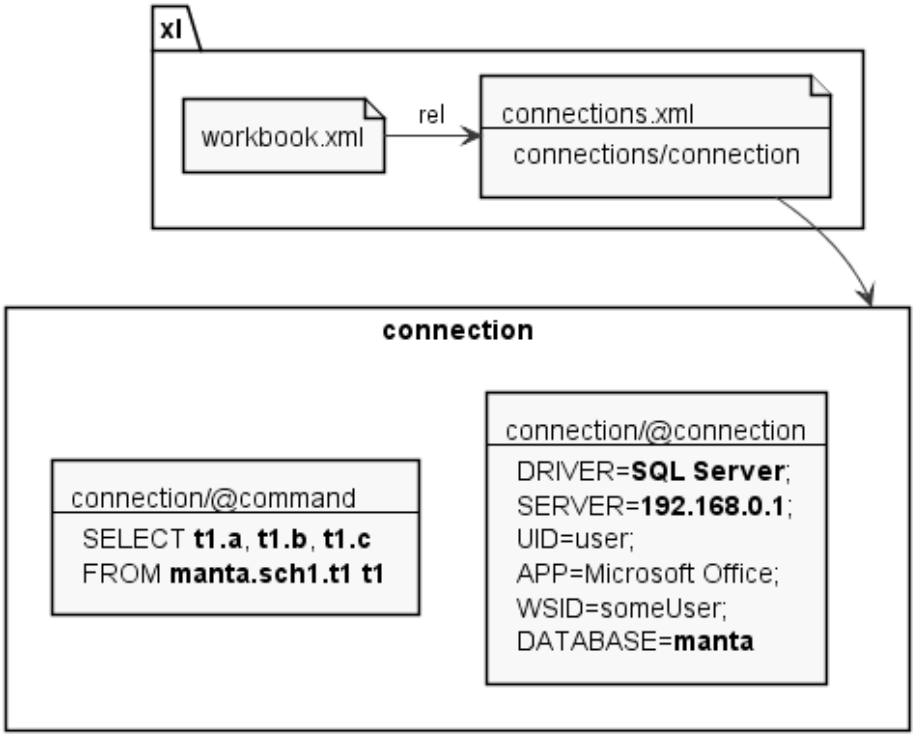
Jednotlivé záznamy o externích datových spojeních sešitu jsou ukládány v souboru `/xl/connections.xml`. Důležitými vlastnostmi záznamů externích spojení jsou atributy `connectionString`, které popisují vlastnosti připojení k externímu zdroji, a `command`, které obsahují příkaz dotazu vracející požadovaný výsledek. [18]

Od verze Microsoft Office 2010 je k dispozici rozšíření Excelu Power Query, které v nejnovější verzích nahrazuje Microsoft Query jako výchozí nástroj pro manipulaci s externími dotazy. Přestože je v současných verzích Microsoft Office nástroj Power Query součástí distribuce, v původní specifikaci OOXML chybí a jeho data jsou ukládána jako vlastní XML soubor obsahující data ve formátu Query Definition File Format `MS-QDEFF`. [19] [20]

Soubor Query Definition File Format je xml soubor obsahující element DataMashup, jehož obsahem je binární stream zakódovaný v base64. Uvnitř streamu jsou zakódované čtyři soubory, z nichž nejdůležitější pro analýzu datových toků je archivní soubor Package Parts ve formátu OPC obsahující mimo jiné soubor `/Formulas/Section1.m`. Ten navíc obsahuje definice všech Power Query M dotazů uložených v daném sešitě.

Uživatel také může zvolit přidání dotazu do datového modelu rozšíření Power Pivot. Při využití této možnosti jsou ze záznamů spojení v souboru `/xl/connections.xml` ztraceny informace o příkazu pro výběr dat v atributu `command`. Datový model rozšíření Power Pivot je ukládán v souboru

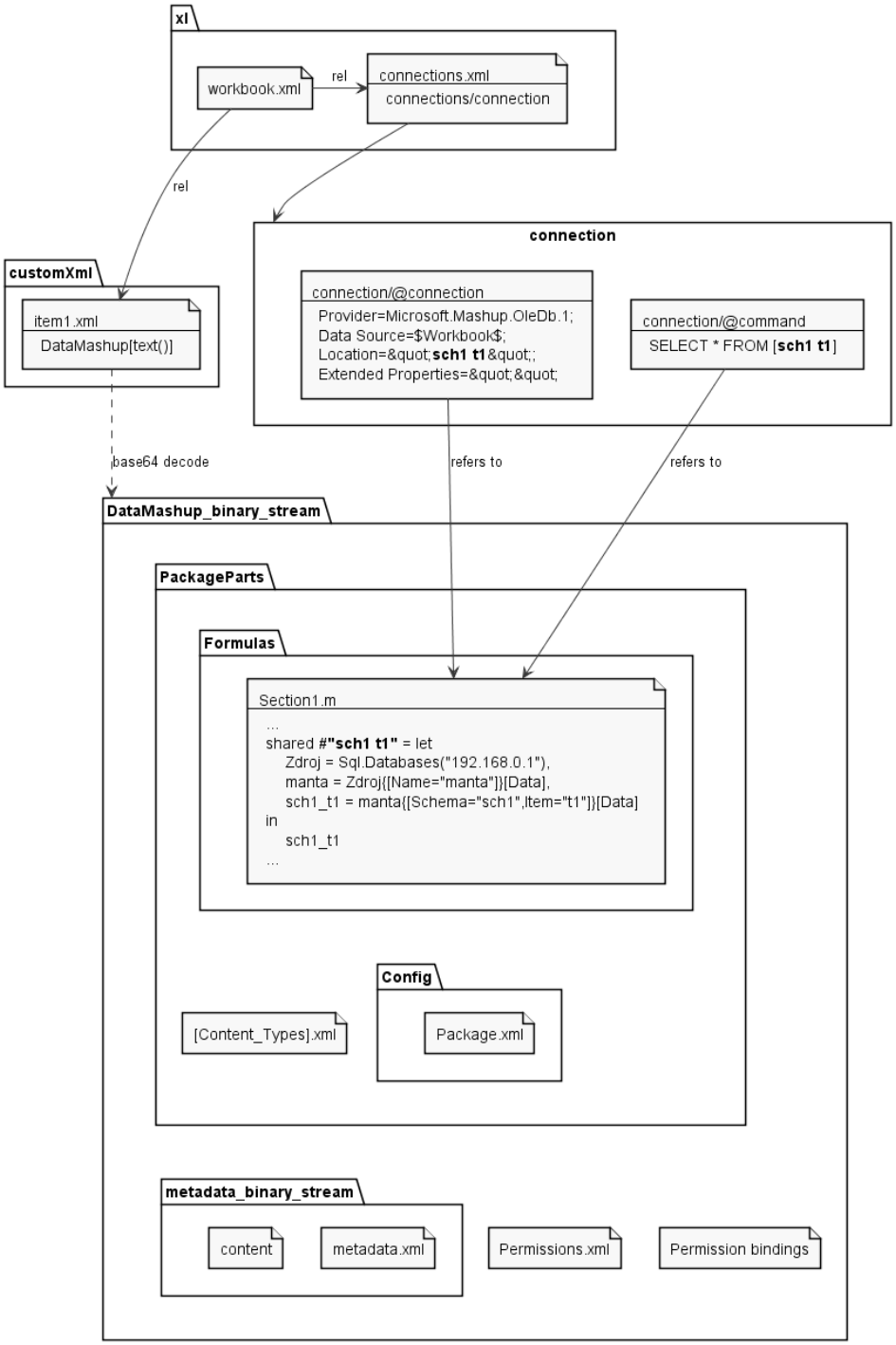
`/xl/model/item1.data` jako bytový stream popsán ve specifikaci `MS-XLDM`. Vnitřní soubory tohoto proudu jsou komprimovány pomocí algoritmu Xpress Compression Algorithm (XCA). Nejdůležitějším z vnitřních souborů je soubor typu `DataSourceViewTabularModel` s názvem `Sandbox`, protože obsahuje ztracený příkaz dotazu. Ostatní funkčnost datového modelu není v prototypu podporovaná, protože je jako rozšíření Excelu nad rámec zadání prototypu. [21] [22]



Obrázek 4.3: Diagram externího dotazu použitím Microsoft Query

4.1.4 Data externího sešitu

V případě referencování dat z jiného sešitu je vytvořen záznam definovaný v souboru `/xl/externalLinks/externalLink1.xml`. V záznamu dat externího sešitu jsou vyjmenovány názvy jednotlivých listů a pro listy, na jejichž buňky jsou reference cíleny, jsou uchovávána data buněk. Samotná adresa sešitu, ze kterého data pochází, se nachází v souboru vztahů daného definičního souboru záznamu dat externího sešitu. Tato adresa je relativní k adrese souboru současného sešitu.



Obrázek 4.4: Diagram externího dotazu použitím Power Query

4.1.5 Definované názvy

Definovaný název je popisný text obsahující validní Excelovský vzorec. Ve vzorci definovaného názvu se mohou vyskytnout tyto konstrukty:

- konstantní hodnoty
- reference na buňku nebo rozsah buněk
- reference na jiný definovaný název
- strukturované reference
- aritmetické operace nebo funkce

Jejich použitím lze zpřehlednit komplikované vzorce nahrazením části kódu vzorce či adresu buňky přehledným zástupným názvem. Jednotlivé názvy jsou přístupné napříč celým sešitem nebo mohou být limitované pouze k určitému listu. Kromě nepovolených znaků není také možné zadefinovat název tvaru adresy buňky v rozsahu `A1-XFD1048576`. Avšak v případě, že uživatel zadefinuje název kolidující s adresou sloupce (jedno až tři velká písmena), je dána přednost definovaným názvum před adresou.

Definované názvy jsou uchovávany v souboru sešitu `/xl/workbook.xml`. Kromě ručního vytvoření se definované názvy vytvářejí i automaticky při vytvoření Excelovské tabulky. [23]

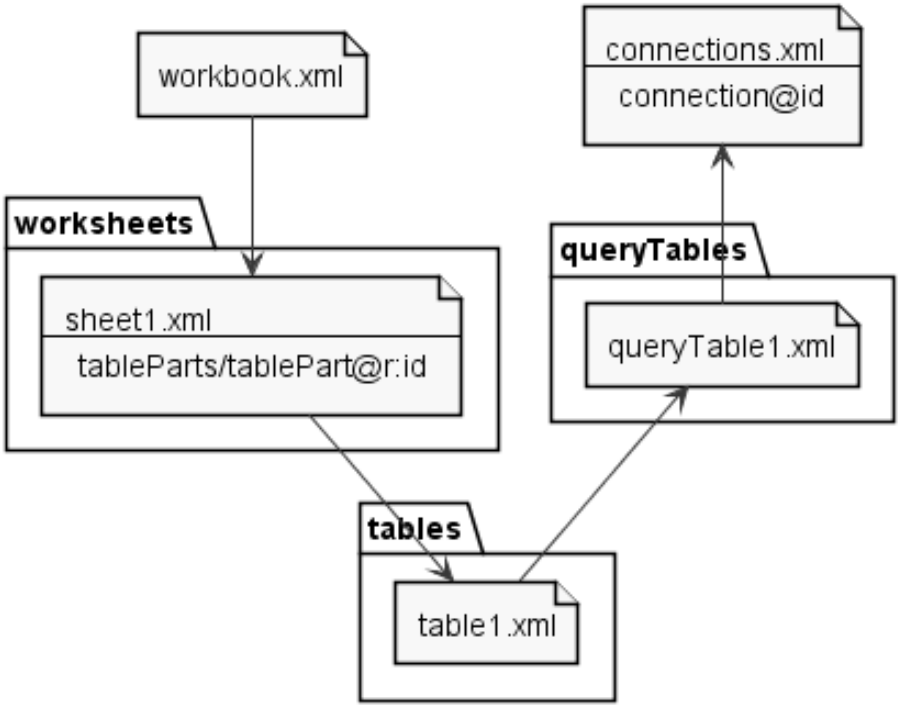
4.1.6 Nativní tabulky

Tabulky jsou explicitně definované objekty Excelu organizující data ve sloupce. Pro každou importovanou tabulku je vytvořen i definovaný název v souboru sešitu odkazující se na rozsah buněk zabíraný tabulkou. Jednotlivé instance tabulek sešitu jsou definované v souboru `/xl/tables/table1.xml` spolu s jejich sloupci a odkazem na dotazovou tabulku, pokud byla tabulka vytvořena importováním z externího zdroje. Obsah buněk tabulky však zůstává v souboru listu sešitu `/xl/worksheets/sheet1.xml`, čímž je umožněno tabulku převést na oblast vyplněných buněk. [24]

Vytvoření tabulky lze provést dvěma způsoby. Prvním z nich je ruční označení buňek a kliknutí na tlačítko Tabulka v záložce Vložení v menu. Tímto vznikne formátovaná oblast tabulky, která buď pojme existující data v oblasti vzniklé tabulky, nebo je uživatel doplní až po vytvoření.

Druhým způsobem je načtením dat na záložce Data a vložením načtených dat do listu sešitu. Při použití tohoto způsobu se automaticky vytvoří i dotazová tabulka spojující vytvořenou tabulku s objektem externího spojení.

Ve vzorcích se lze na sloupce tabulky odkazovat nejen pomocí standardních referencí na adresy buněk, ale i pomocí strukturovaných referencí. Názvy tabulek musí být v celém sešitu unikátní a zároveň musí být unikátní i mezi definovanými názvy.



Obrázek 4.5: Diagram vztahů tabulek a dotazových tabulek

4.1.7 Dotazové tabulky

Dotazová tabulka je dvojrozměrná tabulka, pomocí které je propojeno spojení s externím zdrojem a Excelovská tabulka. Definovaná je uvnitř souboru `/xl/queryTables/queryTable1.xml`. Pokud v sešitě existuje, pojí se právě s jednou Excelovskou tabulkou. S dotazovou tabulkou nelze prostřednictvím Excelu samostatně manipulovat. Pokud je z Excelovské tabulky, která je spojená s dotazovou tabulkou, smazán sloupec, pak je v definičním souboru dané dotazové tabulky tento sloupec přesunut mezi smazané sloupce.[25]

4.1.8 Statické tabulky

Statické či dedukované tabulky jsou obdélníkové oblasti vyplněných buněk listu daty, které uživatel ručně zadal. U tohoto typu dat nelze jednoduše určit homogenitu polí, jejich normální formu a zda-li se jedná tabulku, kde atributy jsou sloupce či řádky. Samy o sobě nemají vlastní soubor s definicí, a proto je nutné implementování vlastního algoritmu pro jejich vyhledávání z buněk v definičním souboru listu sešitu `/xl/worksheets/sheet1.xml`.

4.1.9 Kontingenční tabulky

Kontingenční tabulky jsou další explicitně definované objekty vyskytující se v Excelu. Pomocí nich je možné zobrazit vzájemný vztah více polí jedné zdrojové tabulky. Kontingenční tabulky mohou být výsledkem importu z externího vícedimenzionálního zdroje, kterým je například kostka Microsoft SSAS. Jejich definičním souborem je `/xl/pivotTables/pivotTable1.xml`.

Kontingenční tabulky mohou obsahovat čtyři typy polí:

- sloupce
- řádky
- filtry
- hodnoty

Pole typu sloupců a typu řádků rozřazují agregační hodnoty v datové oblasti podle svých hodnot. Pokud je zvoleno více polí jednoho typu, jsou pole daného typu hierarchicky seskupeny podle pořadí, v jakém jsou zvoleny.

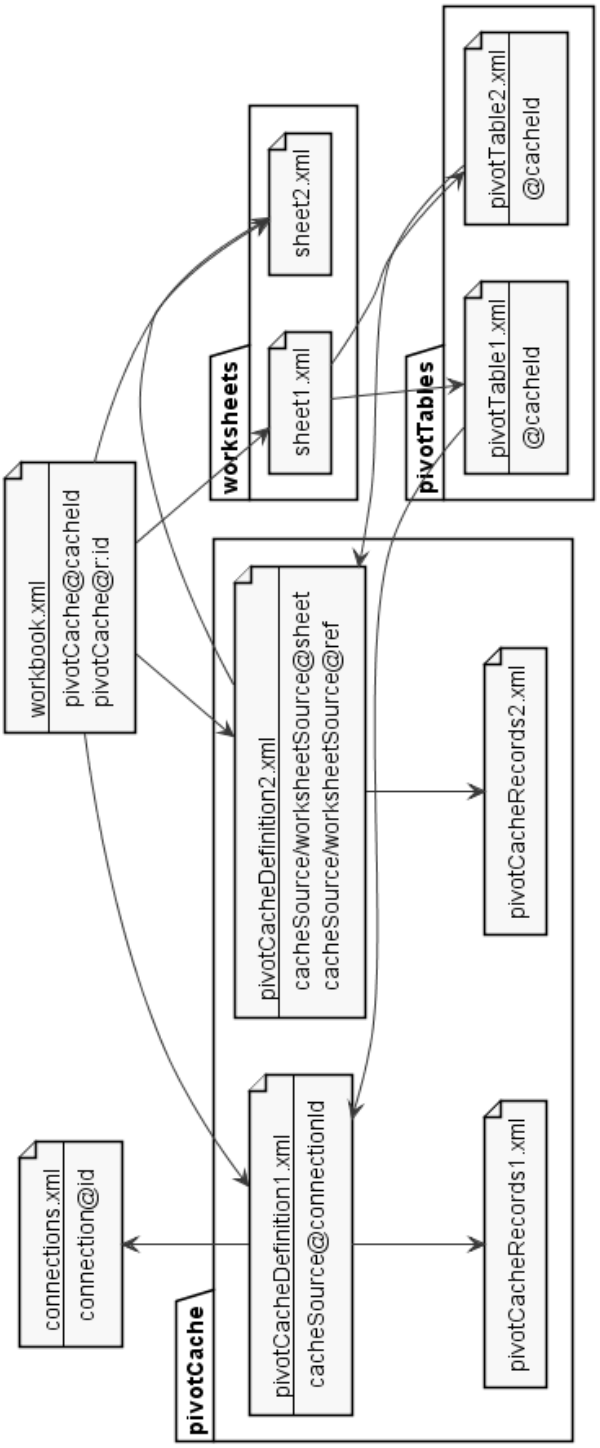
Pole typu hodnot označují pole, jehož hodnoty budou agregovány podle hodnot sloupců a řádků. U těchto sumárních hodnot je také možné vybrat operaci, pomocí které se data agregují (například součet, průměr nebo maximum). Při zvolení více polí hodnot se mezi sloupci objeví nové pole, které udává, o které agregační hodnotě se jedná. Toto pole nelze při zvolení více hodnot smazat a lze jej dále pouze přemístit mezi řádky.

Posledním druhem polí jsou filtry. Pole, které je zvoleno jako filtr, se objeví nad kontingenční tabulkou s mezerou jedné buňky a vedle něj je vytvořena buňka s rozbalovací nabídkou výběru hodnot filtru pole.

Data kontingenčních tabulek jsou vnitřně ukládána do cache, kde mohou být sdílena mezi vícero instancemi. Zdrojem dat cache může být buď oblast v současném či externím sešitu, nebo spojení k externímu zdroji. Při změně zdrojových dat nedojde k aktualizaci cache automaticky a je nutné cache a výslednou kontingenční tabulku aktualizovat ručně.

Tato cache je v souboru sešitu ukládaná ve dvou souborech. Definice cache je ukládána v souboru `/xl/pivotCache/pivotCacheDefiniton1.xml`. Soubor definice cache obsahuje informace o zdroji dat, ze kterého byla data získána. Kromě zdroje dat jsou v definičním souboru cache i záznamy jednotlivých polí dat cache. Pokud jsou tato pole použita v některé z kontingenčních tabulek sešitu, obsahují elementy těchto polí i elementy možných unikátních hodnot daného pole.

Záznamy dat cache kontingenčních tabulek jsou ukládána samostatně uvnitř souboru `/xl/pivotCache/pivotCacheRecords1.xml`. [26]



Obrázek 4.6: Diagram vztahů kontingenčních tabulek

4.1.10 Grafy

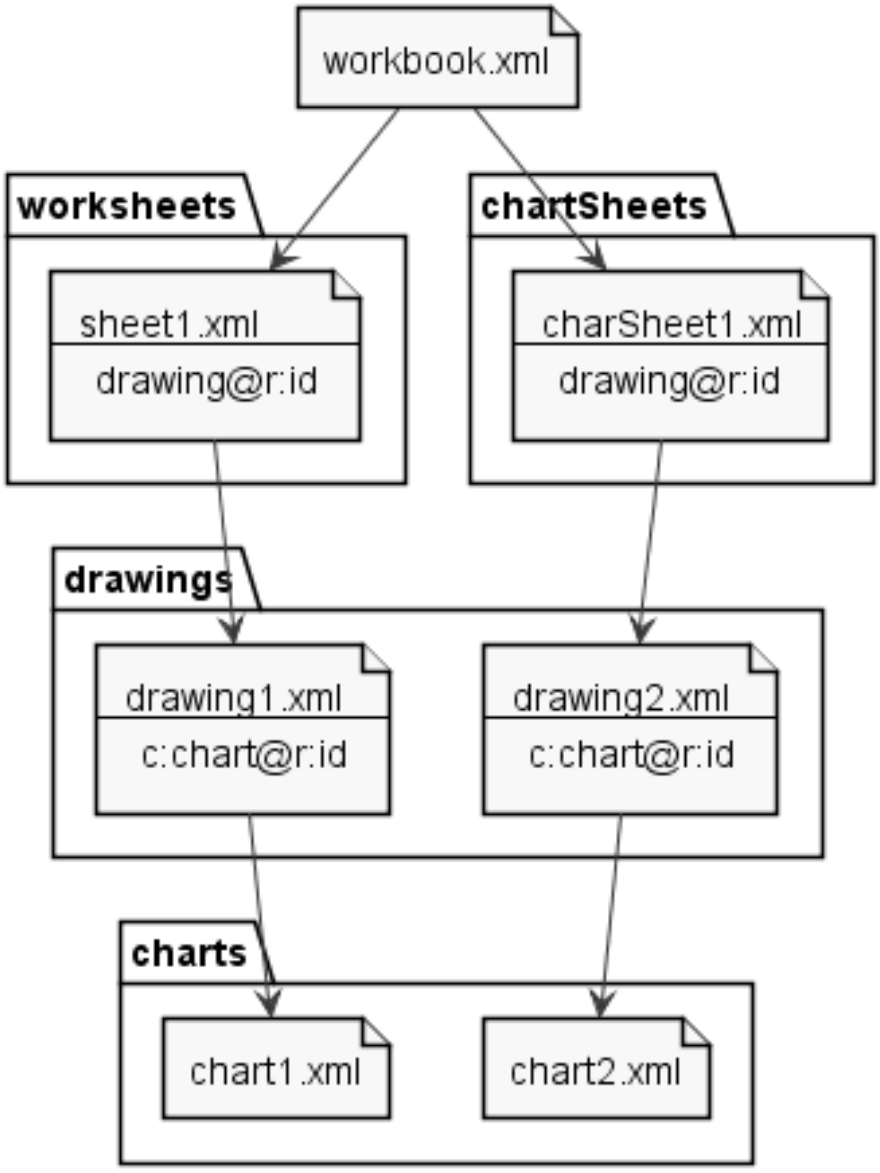
Grafy umožňují vizualizovat data v datových řadách tabulek. K zobrazení využívají specifikace DrawingML. Jednotlivé datové řady grafu se skládají z reference na název řady, reference na kategorie řady a reference na vynášená data. Vyjímkou jsou datové řady bodového grafu, které mají místo referencí na kategorie a data, reference na data osy x a data osy y. Grafy jsou ukládány v definičním souboru `/xl/charts/chart1.xml`.

Typ grafu se vztahuje k samotným datovým řadám grafu, a ne grafu jako celku. Z toho důvodu lze vytvořit jeden graf s datovými řadami navzájem odlišnými, což se v Excelu nazývá kombinační graf.

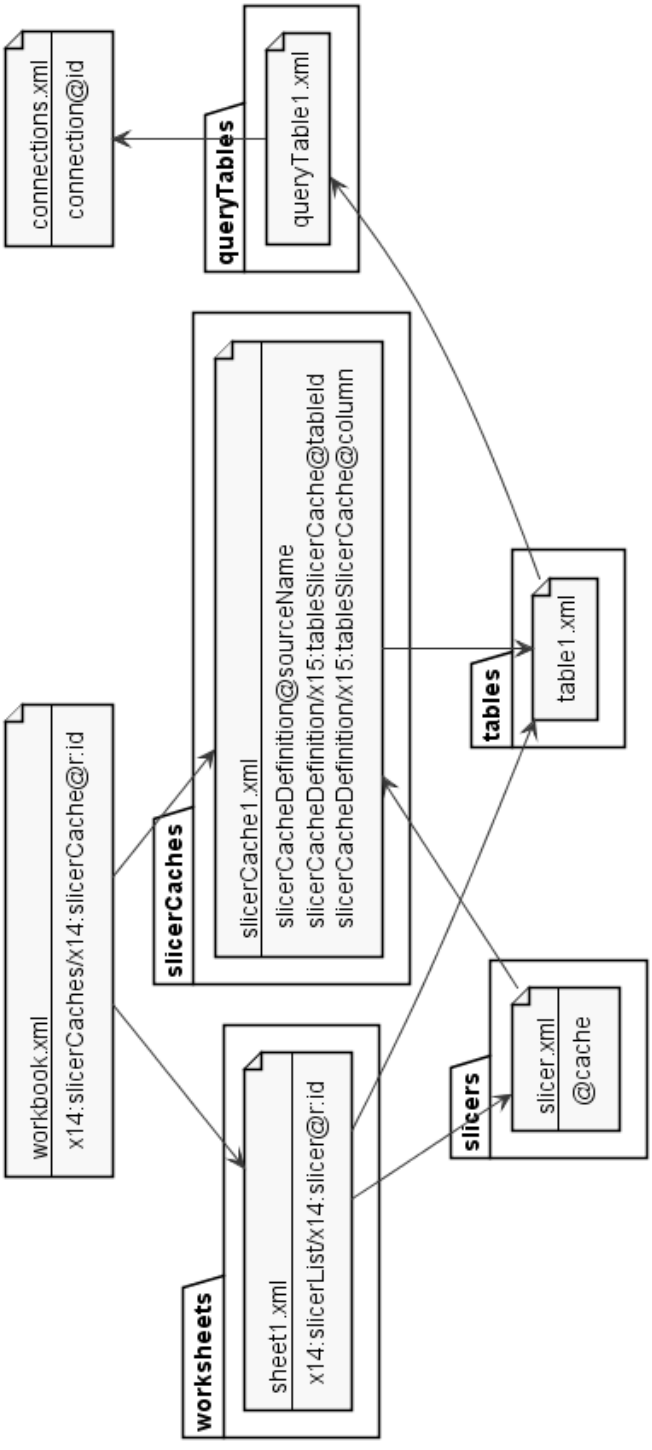
Při vytváření grafu lze jako zdroj dat vybrat oblast dat, tabulku nebo kontingenční tabulku. Graf je možné přesunout na vlastní druh listu, který je definovaný zvlášť od standardních listů v `/xl/chartSheets/chartSheet1.xml`. Ve specifikaci grafu je dále spousta možností přizpůsobení vzhledu, které však nejsou součástí toku dat, a proto jsou ignorovány. [27]

4.1.11 Průřezy

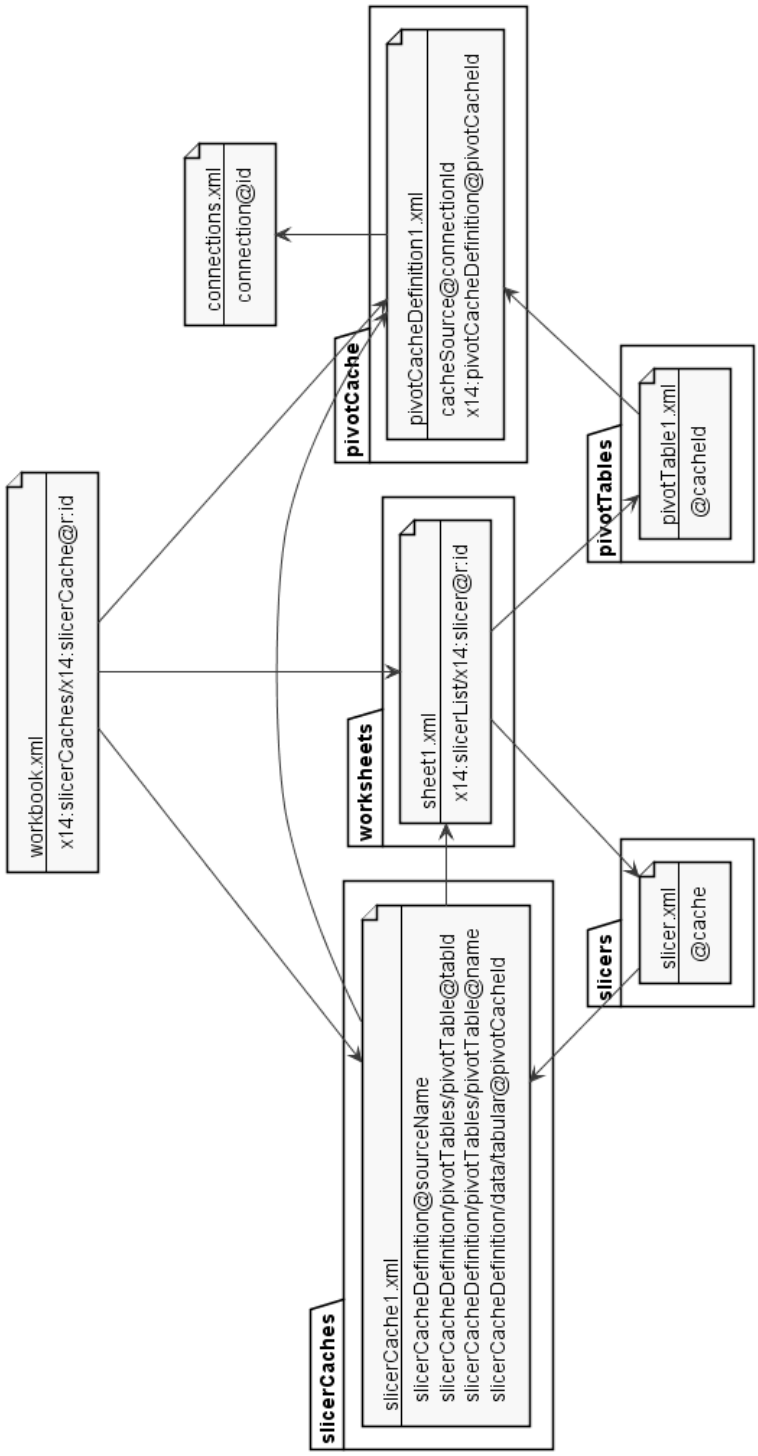
Průřezy jsou filtrovací objekty vykreslované pomocí DrawingML na Excelovském listu, které jsou definovány v `/xl/slicers/slicer1.xml`. V přidruženém objektu `/xl/slicerCaches/slicerCache` jsou uchovávány informace o datech tabulky nebo kontingenční tabulky, kterou filtrují. Každý průřez může filtrovat hodnoty pouze jednoho pole zvolené tabulky. Průřezy jsou objektem, který byl do Excelu přidán od verze 2010, kdy mohl filtrovat pouze kontingenční tabulky. Od verze 2013 lze pomocí průřezů filtrovat i obyčejné Excelovské tabulky. [28]



Obrázek 4.7: Diagram vztahů grafu



Obrázek 4.8: Diagram průřezu filtrujícího tabulku



Obrázek 4.9: Diagram průřezu filtrujícího kontingenční tabulku

4.2 Externí dotazy

Cílem analýzy datových toků je spojit reporty vytvořené v Excelu s externím zdrojem dat, ze kterého byla data importována. Nejnovější verze Excelu používají pro tvorbu dotazů nástroj Power Query jako výchozí, avšak stále zůstává možnost použít starší nástroj Microsoft Query. Tyto nástroje budou v této sekci představeny.

4.2.1 Microsoft Query

Microsoft Query je vestavěný nástroj pro tvoření dotazů a získávání dat z externích zdrojů umožňující automatické aktualizování při změnách v originálním zdroji. Mezi předinstalované ovladače přístupu k externím zdrojům pro Microsoft Query patří jak ovladače pro nástroje od firmy Microsoft (např. Microsoft Office Access, Microsoft Office Excel, Microsoft SQL Server, Microsoft SQL Server Analysis Services), tak ovladače cizích technologií jako jsou dBASE nebo Oracle. Kromě předinstalovaných ovladačů umožňuje Microsoft Query použít ODBC[9] či jiné ovladače.

Po vytvoření spojení ke konkrétnímu zdroji je možné vytvořit více dotazů bez nutnosti vytvořit nové, duplicitní spojení. Excel ukládá jak spojení, tak dotaz, čímž umožňuje opakované připojování. Díky tomu je možné udržovat aktuální jak data, tak objekty na těchto datech závislých. Pro tvorbu dotazů je použit nástroj Query Wizard umožňující tvořit dotazy jak vizuálním spojováním objektů, tak psaním dotazů dotazovacím jazykem, který daná technologie používá.[29]

4.2.2 Power Query a Get & Transform

Od verze 2010 je k dispozici nástroj Power Query jako alternativní způsob manipulace s dotazy na externí zdroje. Power Query vznikl jako jedna ze součástí sady rozšíření Excelu s názvem Power BI. Dalšími součástmi této sady byl interaktivní vizualizační nástroj Power View a nástroj pro modelování dat Power Pivot. Od verze 2013 určité varianty balíku Microsoft Office (konkrétně Office Professional Plus 2013 a Office 365 Pro Plus) obsahovaly předinstalované nástroje Power Query, Power Pivot a Power View. [30] Ve verzi 2016 se nástroj Power Query přejmenoval na Get & Transform (v české lokalizaci jako Načíst a transformovat) a dodával se předinstalovaný se všemi verzemi balíku Microsoft Office a Microsoft Office 365.[31] [32]

Kromě vizuálního rozhraní pro úpravu dotazu k externímu zdroji přináší nástroj Power Query vlastní dotazovací jazyk M. Stejně jako grafické rozhraní sestavování dotazu, i dotaz v jazyku M je tvořen jednotlivými transformačními kroky, kde v každém kroku je nové proměnné přiřazen výsledek transformace předchozí proměnné. [33]

[9] Open Database Connectivity

4.3 Datové toky

Pro analýzu datových toků je nutné analyzovat jednotlivé reference a spojit zdrojový uzel s cílovým. Datové toky mohou být dvou druhů. Prvním z nich je přímý datový tok, který indikuje, že daný uzel se přímo podílí na obsahu jiného. Druhým druhem je nepřímý tok, který označuje, že uzel filtruje výsledná data jiného.

Pro generování vizualizace datových toků mezi objekty uvnitř sešitu je potřeba parsovat výrazy referencí, kterými na sebe objekty odkazují. Tyto reference používají jazyk Excelovských vzorců, který umožňuje referovat jak na jiné elementy v rámci sešitu, tak na tabulky nebo datové sady z externích zdrojů.

Podoba importovacích dotazů z vnějších zdrojů se liší v závislosti na tom, jestli k vytvoření dotazu byl použit nástroj Microsoft Query nebo Power Query. V případě použití Microsoft Query je celý dotaz definován v jazyku technologie zdroje externích dat. Nástroj Manta Flow obsahuje parsery dotazovacích jazyků technologií, které podporuje. Z tohoto důvodu není nutné tyto jazyky zmiňovat.

Pokud však uživatel k vytvoření dotazu použije Power Query, je k definování dotazu použit dotazovací mashup jazyk Power Query M. Množina všech mashup dotazů definovaná v sešitě je pak použita jako externí zdroj pro dotaz nástroje Microsoft Query, ze kterého jsou data konkrétního dotazu získány pomocí jednoduchého SQL výrazu, kde v klauzuli `SELECT` se nachází zástupný znak a v klauzuli `FROM` název definovaného Power Query M dotazu v hranatých závorkách.

4.3.1 Jazyk Excelovských vzorců

Excel obsahuje vestavěný jazyk pro zapisování výpočtu hodnot buněk pomocí vzorců. Vzorce mohou obsahovat výrazy obsahující konstatní hodnoty, aritmetické operace, funkce, definované názvy či reference na buňky. [34]

Reference tvoří důležitou součást velkého procenta všech vzorců. Při změně hodnoty referované buňky jsou buňky, které na danou buňku referovaly vzorcem, přepočítány s ohledem na její současnou hodnotu. Běžné buněčné reference jsou tvořeny její adresou v notaci `A1`. Reference v tomto tvaru je relativní a při přesunutí buňky se adresa odkazované buňky změní se stejným odsazením jako odkazující buňka. Pro specifikování absolutní reference je nutné zapsat před požadovanou řádkovou nebo sloupcovou souřadnicí symbol dolaru: `=$A$1`.

Kromě samotných referencí obsahuje Excel i operace rozsahu adres buněk. Samotný rozsah buněk je jedna z klíčových součástí vzorců a značí se jako dvě adresy buněk indikující začátek a konec rozsahu. Pro oddělení těchto dvou referencí se používá symbol dvojtečky. Reference rozsahu tak může vypadat například jako `=A1:B2`. Alternativně je možné zapsat rozsah všech buněk dvou

sloupců nebo dvou řádků jako adresu počátečního a konečného sloupce či řádku respektivně. Takto zapsané reference mají podobu `=A:C` nebo `=1:3`. Rozashy zapsané tímto způsobem jsou ekvivalentní rozsahům `=A1:C1048576` a `=A1:XFD3` respektivně.

Mimo standardních rozsahových referencí obsahující pouze počáteční a konečnou referenci je možné použít rozsahy o více členech. Při vyhodnocení takového rozsahu je rozsah nahrazen ekvivalentních rozsahem dvou referencí, kde první obsahuje minimální řádek a sloupec původního rozsahu a druhá zase maximální řádek a sloupec. Příkladem tohoto rozsahu může být vzorec `=B2:D3:C4`, který je ekvivalentní vzorci `=B2:D4`. Rozsah lze rozšířit pomocí operace sjednocení, která se interně a v anglické lokalizaci značí jako čárka: `=A1:B2,C3:D4`. V české lokalizaci je tento symbol nahrazen středníkem. Tato operace není příliš využívaná, jak vychází z tabulky četnosti výskytu jednotlivých konstruktů [34], kde procentuální výskyt tohoto jevu byl nižší než setina procenta. Tento fakt si lze vysvětlit tím, že operace sjednocení využívá totožný znak operátoru jako oddělovač argumentů funkcí. Vzorec zapsaný jako `=SUM(A1:B2,C3:D4)` je pak interpretován jako funkce se dvěma argumenty, místo jednoho argumentu tvořeným sjednocením buněk. Výskyt sjednocení tak bývá častý pouze u funkcí, které vyžadují rozsah buněk jako jediný argument. Těmito funkcemi jsou například `LARGE`, `SMALL` nebo `RANK`. Přestože se tato operace označuje jako sjednocení rozsahu buněk, jde ve skutečnosti o množinový součet. Toto lze ověřit použitím funkce `COUNT` a sjednocením dvou identických rozsahů buněk, čímž se dostaneme k výsledku, který bude dvojnásobný od očekávaného.

Poslední množinovou operací je průnik, který se značí mezerou: `=A1:B2 B1:B3`. Průnik je podobně jako sjednocení málo využívaným nástrojem jazyka Excelovských vzorců. Tuto skutečnost dokládá tabulka [34], jejíž hodnoty ukazují menší než jednoprocentní výskyt.

Reference ve vzorci buňky může odkazovat nejen do listu, kde se buňka se vzorcem nachází, ale i do ostatních listů. Tyto externí reference se značí prefixem s názvem listu, který je oddělený vykřičníkem: `=List1!A1`. Pokud název listu v externí referenci obsahuje mezeru nebo jiné znaky nepovolené ve vzorcích, je ohraničen pomocí apostrofu: `='List 1'!A1`.

Další možností externí reference je odkaz na buňky, které se nachází v externím sešitu. Sešit cíle reference objektu externího souboru se podobně jako reference na objekty jiného listu zapisují před referovaný objekt, který je oddělený vykřičníkem. Uvnitř aplikace se adresa externího sešitu zobrazuje jako název souboru sešitu v hranatých závorkách, který je předcházen absolutní cestou k souboru: `=C:\Users\user\[Workbook.xlsx]List1!A1`. Pokud je tento sešit zároveň otevřený v jiném okně aplikace, zobrazuje se pouze jeho název souboru v hranatých závorkách: `=[Workbook.xlsx]List1!A1`. Interně se ale odkazy na externí soubory ukládají jako číslo v hranatých závorkách, které značí index záznamu v externích sešitech obsahující adresu souboru. Takto může vypadat odkaz na buňku v jiném sešitu: `='[1]List 1'!A2`.

Kromě externích referencí na buňky, které specifikují externí sešit i list, je možné referencovat definované názvy externích sešitů, které jsou sdílené všemi listy daného sešitu, proto je v referenci specifikovaný pouze sešit. Příkladem externí reference na definovaný název může být vzorec: `=[1]!Constant`.

Excel také umožňuje pro externí reference použít operátor rozsahu, čímž je možné kalkulovat s oblastí buněk stejné adresy na vícero listech. Příklad takto zapsaného rozsahu ukazuje vzorec: `=List1:List3!A1`. Tento konstrukt však není příliš používán, jak vyplývá ze statistiky vypracované v [34].

Při použití tabulek v Excelovském sešitu je možné na ně odkazovat pomocí strukturovaných referencí. Strukturované reference se skládají z názvu samotné tabulky a specifikací položky a sloupců tabulky, které jsou zapisovány v hranatých závorkách. Specifikací položky je možné vybrat všechna data (pomocí `#All`), jen hlavičky (pomocí `#Headers`), jen data (pomocí `#Data`), součty (pomocí `#Totals`) nebo buňky ve stejném řádku jako buňka obsahující tento vzorec (pomocí `#This Row` nebo zkráceně pomocí `@`). Při vynechání specifikace položky se jako výchozí použijí všechna data. Výběr sloupců je možný buď jednotlivě, nebo pomocí operátoru rozsahu. Výsledná reference tak může vypadat například: `=Tabulka1[[#Data],[Sloupec1]:[Sloupec3]]`. Stejně jako reference na adresy buněk, tak i strukturované reference mohou obsahovat externí část specifikující sešit, kde se odkazovaná tabulka nachází. [35]

Mimo explicitně specifikovaných referencí ve vzorci lze použít i zástupné proměnné zvané definované názvy. Jak už bylo zmíněno v 4.1.5, obsah definovaných názvů je jakýkoliv validní Excelovský vzorec. Obsahovat může konstantní hodnoty, reference, rozsahy referencí, funkce či reference na jiné definované názvy. Definované názvy nesmí být pojmenovány stejně jako adresy buněk či existující tabulky, avšak je možné pojmenovat definované názvy kombinací tří velkých písmen, čímž kolidují s názvy sloupců. Jediné místo, kde je možné použít samotné sloupcové reference, je v rozsahu sloupců. Pokud by nastal případ, že v sešitu existuje rozsah dvou sloupců, a zároveň existuje definovaný název se stejným názvem jako jeden ze sloupců rozsahu, pak přednost ve vyhodnocování dostane sloupec.

Excel v současnosti obsahuje 477 vestavěných funkcí. [36] Uživatelé mohou definovat vlastní UDF (User defined function) pomocí jazyku Visual Basic for Applications. Názvy vestavěných funkcí obsahují pouze velká písmena, čísla nebo tečky. Toto ovšem není omezení pro uživatelem definované funkce, které mohou obsahovat i malá písmena. Argumenty funkcí se zapisují do kulatých závorek a jsou odděleny čárkou (v české lokalizaci středník). Jednotlivé argumenty mohou mít odlišné druhy datových toků v závislosti na tom, jestli se daný tok přímo nebo nepřímo podílí na vypočtené hodnotě výsledného pole. [34] [37]

4.3.2 Jazyk Power Query M

Power Query M nebo také Power Query Formula Language je funkcionální jazyk podobný F# určený k vytváření data mashup dotazů v nástrojích Microsoft Excel a Microsoft Power BI Desktop. Účelem jazyku je filtrování a kombinování dat z různých podporovaných zdrojů dat, čímž je vytvářen tzv. "data mashup". Jako kořenový element jazyka M je dokument, který je tvořen buď pouze jedním výrazem, nebo skupinou definicí organizovaných do sekcí.

Každá definice dotazu začíná klíčovým slovem `let`. Dotaz jazyka M se podobně jako dotaz sestavovaný pomocí uživatelského rozhraní skládá z jednotlivých kroků. Tyto kroky jsou tvořeny jako přiřazení hodnoty nové krokové proměnné pomocí výrazu, který manipuluje s předchozí proměnnou. Pro oddělení kroků jsou použité čárky. Identifikátory proměnných mohou kromě alfanumerických znaků obsahovat i mezery, přičemž název proměnné je ohraničen uvozovkami a předcházen znakem #. Kroková proměnná, jejíž hodnota je považována za návratovou je na konci po deklarování specifikována pomocí klíčového slova `in`.

Jednotlivé proměnné mohou být označené za sdílené pomocí klíčového slova `shared`, čímž je na ně umožněno referovat z jiných sekcí bez jejich explicitního specifikování, pokud jsou mezi sdílenými proměnnými unikátní.

Power Query M obsahuje velké množství vestavěných funkcí získávajících data z různých externích zdrojů. Pro použití zdrojů, které nejsou nativně podporovány lze využít funkce obecných konektorů OLE DB[10] a ODBC. Tyto funkce zpřístupňují hierarchicky nejvyšší element daného datového zdroje, typicky databázový server nebo konkrétní databázi. Tento element má podobu navigační tabulky, kde v prvním sloupci jsou názvy položky a v druhém její obsah.

Pro zpřístupnění určité položky lze využít dva druhy operátoru výběru. Prvním z nich je vyhledávací operátor, který se zapisuje hranatými závorkami a obsahuje název položky, která má být zpřístupněna. Název položky může být jednoduchý či složený z více identifikátorů oddělených čárkou. Druhou možností je operátor indexu pozice, který je zapisovaný jako složené závorky s číselným indexem požadované položky uvnitř. Číselné indexy v Power Query M začínají od nuly. [38]

[10] Object Linking and Embedding, Database

Kapitola 5

Návrh

5.1 Datový model

V této kapitole bude představen návrh rozhraní datového modelu Excelu. Podoba datového modelu se blíží podobě struktury uložených dat. Zmíněna budou důležitá rozhraní jednotlivých objektů datového modelu. Kompletní podoba hierarchie datového modelu je vyobrazena na diagramu 5.1.

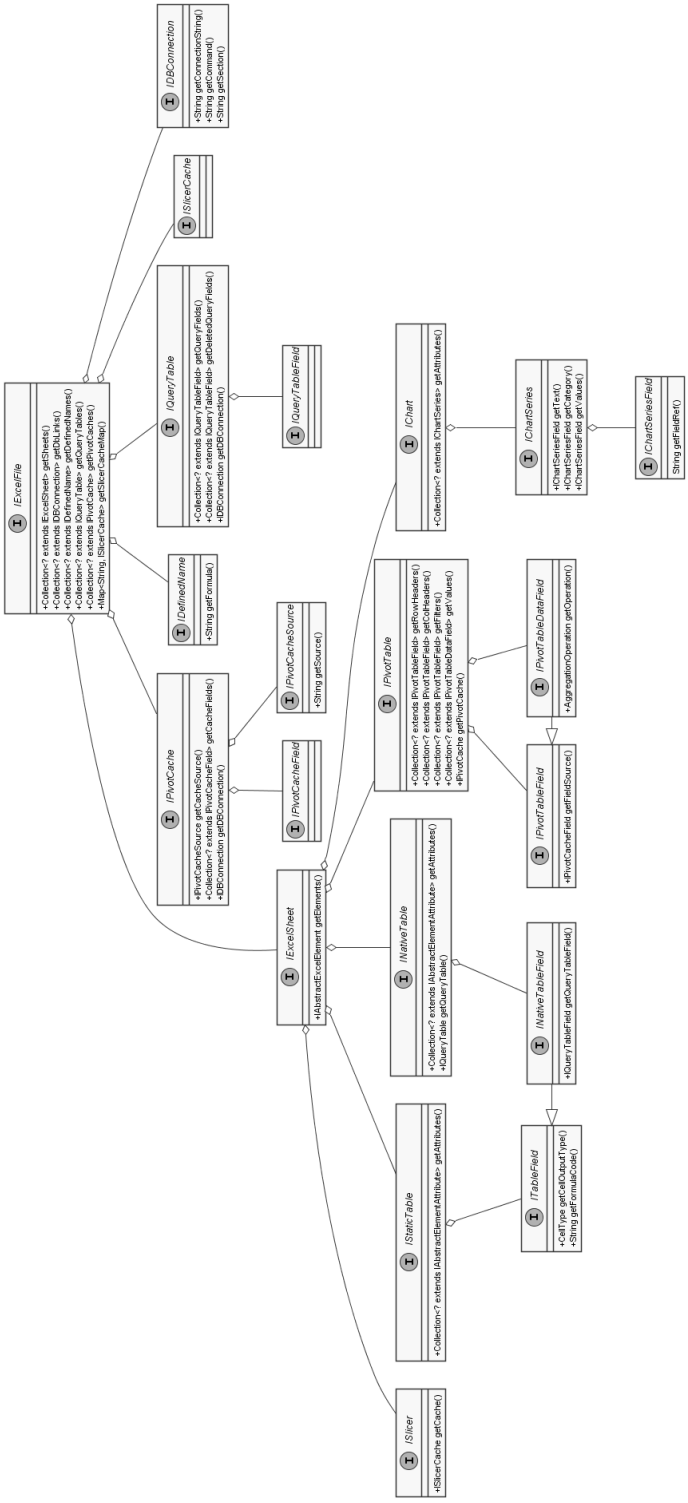
5.1.1 IExcelWorkbookManager

`IExcelWorkbookManager` je rozhraní zpřístupňující jednotlivé sešity pomocí metody `getExcelFile(String fileName)`, čímž tvoří určitý kontext. Díky tomuto objektu je možné získat objekty referencované napříč sešity. Tímto je nezbytný pro vytváření vizualizace datových toků.

5.1.2 IExcelWorkbook

Rozhraní `IExcelWorkbook` reprezentuje Excelovský sešit, čímž reprezentuje kořenový element analyzovaného souboru sešitu. Tento objekt obsahuje metody, které zpřístupňují listy `getSheet(String sheetName)` a `getSheets()`. Kromě sešitů zpřístupňuje i objekty sdílené v celém sešitu. Jmenovitě se jedná o tyto objekty:

- odkazy na externí sešity (pomocí metody `getExternalLink(int id)`)
- definované databázové spojení (pomocí metody `getDbLinks()`)
- definované názvy (pomocí metody `getDefinedName(String key)`)
- dotazové tabulky (pomocí metody `getQueryTables()`)
- cache kontingenčních tabulek (pomocí metody `getPivotCaches()`)
- cache průřezů (pomocí metody `getSlicerCacheMap()`)



Obrázek 5.1: Diagram tříd datového modelu

5.1.3 ExcelNode

Všechny objekty hierarchie sešitu rozšiřují rozhraní `ExcelNode`, které obsahuje metody zpřístupňující rodiče objektu a jeho název.

5.1.4 IExcelSheet

Listy jsou v datovém modelu reprezentovány rozhraním `IExcelSheet`. Metodami `getElement(int id)`, `getElement(String name)` a `getElements()` jsou zpřístupňovány jednotlivé objekty, které se mohou v listech nacházet.

Objekty, které zabírají určitý rozsah buněk v listu, lze zpřístupnit pomocí metod `getTable(int row, int col)` a `getTable(CellAddress ca)`. Tyto metody vrací objekty implementující rozhraní `ICellTable`, kterými mohou být tabulky nebo kontingenční tabulky.

5.1.5 ISheetElement

Obecným rozhraním pro všechny objekty listu je rozhraní `ISheetElement`. Kromě názvu obsahuje metodu `getFields()`, která zpřístupňuje všechny jeho pole, které implementují rozhraní `IElementField`.

Rozhraní `IElementField` rozšiřují specifické rozhraní pro jednotlivé druhy polí. Těmito rozhraními jsou:

- `ITableField` (reprezentující sloupce tabulky)
- `IPivotTableField` (reprezentující pole kontingenční tabulky)
- `IPivotCacheField` (reprezentující pole kontingenční cache)
- `IQueryTableField` (reprezentující sloupec dotazové tabulky)
- `IChartSeries` (reprezentující datovou řadu grafu)
- `IChartSeriesField` (reprezentující pole datové řady grafu)

5.1.6 IDBOrigin

Objekty přímo vzniklé importováním dat z externího zdroje popisuje rozhraní `IDBOrigin`. Tyto objekty mohou být buď dotazová tabulka, která je popsána rozhraním `IQueryTable`, nebo objekt cache kontingenční tabulky, který je popsán rozhraním `IPivotCache`. Pomocí jeho metody `getDBConnections()` lze zpřístupnit konkrétní informace o databázovém spojení, kterým byl objekt vytvořen.

Tyto informace jsou uchovávány v objektu, který je popsaný rozhraním `IDBConnection`. Tento objekt zpřístupňuje kromě společných informací pro nástroje Microsoft Query a Power Query, kterými jsou `connectionString` a `command`, může obsahovat i definici dotazu Power Query.

5.1.7 ICellTable

Rozhraní `ICellTable` popisuje obecnou tabulku nacházející se v mřížce buněk sešitu. Toto obecné rozhraní implementují standardní homogenní tabulky, i kontingenční tabulky. Metody tohoto rozhraní se týkají hlavně lokace tabulky v mřížce buněk. Pomocí metod `getBeginning()`, `getTableWidth()` a `getTableHeight()` lze zjistit rozměry a lokaci tabulky.

Metoda `isInside(int row, int col)` slouží k rychlému ověření, že adresa popisována řádkem a sloupcem se nachází v rozsahu buněk tabulky.

Metoda `isAdjacent(CellAddress cellAddress)` slouží k zjištění a rozhodování o příslušnosti buňky k tabulce. Pomocí ní je při procházení buněk rozhodováno o rozšíření tabulky. Tato metoda vrací `true` se buňka nachází v těsné blízkosti této tabulky.

Pomocí metod `getFields(CellAddress first, CellAddress second)` a `getFields(CellAddress first)` je možné zpřístupnit pole dané tabulky adresami buněk.

5.1.8 IHomogeneousCellTable

`IHomogeneousCellTable` je rozšířením rozhraní `ICellTable`, které reprezentuje homogenní dvojrozměrné tabulky jako jsou statické či nativní tabulky, které jsou reprezentovány rozhraními `IStaticTable` a `INativeTable` respektivně. Jelikož statické tabulky nemusí obsahovat záhlaví polí a tyto pole nemusí mít vždy podobu sloupců, obsahuje toto rozhraní metody `hasHeaders()` a `isPortrait()`.

Pomocí metody `getFields(int first, int second)` a `getField(int index)` je možné přistupovat k polím podle podle jejich indexu, neboť obsahují jen jeden druh polí.

5.1.9 IPivotTable

Rozhraní `IPivotTable` rozšiřuje rozhraní `ISheetElement` a `ICellTable` o metody specifické pro objekt kontingenční tabulky. Data kontingenční tabulky a informace o jejím zdroji jsou uchovávány v kontingenční cache, jejíž referenci lze získat metodou `getPivotCache()`.

Kontingenční tabulka obsahuje čtyři druhy polí a proto její rozhraní deklaruje metody zpřístupňující kolekce každého druhu pole.

- `getRowHeaders()`
- `getColHeaders()`
- `getFilters()`
- `getValues()`

Kromě metod, které zpřístupňují celé kolekce těchto polí jsou v rozhraní přítomné i metody zpřístupňující jedno určité pole daného druhu podle jeho indexu.

Řádky, sloupce a filtry kontingenční tabulky jsou popisovány rozhraním `IPivotTableField`, které obsahuje popisek pole a jeho typ. Dále pak rozhraní pole metodou `getFieldSource` zpřístupňuje referenci na zdrojové pole v kontingenční cache. Pole hodnot kontingenční tabulky je reprezentováno pomocí rozhraní `IPivotTableDataField`, které navíc oproti `IPivotTableField` obsahuje vygenerovaný název daného pole a referenci na agregační operaci, kterou jsou hodnoty tohoto pole agregovány.

5.1.10 IPivotCache

Toto rozhraní reprezentuje cache kontingenční tabulky, která shromažďuje informace o jednotlivých polích zdrojových dat a o jejich původu. `IPivotCache` rozšiřuje rozhraní `IDBOrigin`, jelikož tento typ objektů může být cílem datového toku z externího datového zdroje. Jednotlivá pole cache zpřístupňují metody `getCacheFields()` a `getCacheField(int index)`. Pomocí metody `getCacheSource()` lze zpřístupnit objekt typu `IPivotCacheSource`, který obsahuje referenci na zdrojová data a typ zdroje dat (reference nebo databázové spojení).

5.1.11 IChartSeries

Rozhraní `IChartSeries` popisuje jednotlivé datové řady grafu typu `IChart`. Rozhraní datové řady grafu obsahuje informace o typu grafu pro danou datovou řadu, které lze zpřístupnit pomocí `getType()`.

Každá datová řada obsahuje pole referencí na název řady, kategorii řady (textové popisky či hodnoty na ose X) a hodnoty (hodnoty na ose Y). Tyto pole jsou reprezentována rozhraním `IChartSeriesField` a jsou získávána metodami `getText()`, `getCategory()` a `getValues()` respektivně.

5.1.12 ISlicer

Průřezy jsou reprezentovány rozhraním `ISlicer` a nabízí metodu `getCache()` pro zpřístupnění cache jeho dat, která je popisována obecným rozhraním `ISlicerCache`. Z obecného rozhraní dědí dvě specializovaná rozhraní. Prvním z nich je `INativeTableSlicerCache`, popisující cache průřezu filtrujícího Excelovskou tabulku, a druhým je rozhraní `IPivotTableSlicerCache`, reprezentující cache průřezu filtrující kontingenční tabulku.

5.2 Gramatika parseru Excelovských vzorců

Pro správné určování datových toků objektů referovaných v Excelovských vzorcích je nutné navrhnout gramatiku, pomocí které lze jednoznačně identifikovat reference v textu Excelovského vzorce. Pro vytvoření parseru je použit nástroj ANTLR verze 3.5.

Gramatika byla navrhována postupným rozšiřováním jednoduché aritmetické gramatiky o konstrukty specifické pro jazyk Excelovských vzorců. Jednotlivé iterace gramatiky pak byly testovány na uměle vytvořených příkladech validních funkcí a na testovacích datech projektu XLParser.[39]

Při návrhu bylo dbáno na jednoznačnost gramatiky i na úkor počtu podporovaných konstruktů jazyka. Mezi vynechané konstrukty patří například vektorové či maticové výpočty, které se zapisují pomocí složených závorek. Příkladem takových výpočtů může být například {`=MMULT(E2:G5,A2:C4)`}, pomocí které se maticově vynásobí dvě matice specifikované jejich rozsahem buněk.

Dalším vynechaným konstruktem jsou množinové součty, které jsou označovány jako sjednocení, přestože překrývající regiony nejsou od výsledku odečteny. Důvodem jejich vynechání je jejich nejednoznačnost s ohledem na argumenty funkcí, jenž jsou oddělovány stejným znakem jako je značen množinový součet. Při jejich zahrnutí do gramatiky je nejednoznačné, zda-li vzorec zapsaný jako `=SUMIF(A2:A6,D2,C2:C6)` obsahuje tři argumenty či jeden argument, jímž je množinový součet tří referencí. Pro analýzu datových toků mají přednost samotné argumenty, jelikož podle jejich pořadí lze určit, které argumenty dané funkce mají typ toku přímý a které nepřímý.

Jako další vynechané konstrukty lze zmínit rozsahy napříč listy sešitu a komplexní rozsahy. Jako příklad rozsahů napříč listy lze uvést konkrétní vzorec `=SUM(Sheet1:Sheet3!C5)`, který sečte buňky stejné adresy ze všech listů ve specifikovaném rozsahu listů. Komplexními rozsahy jsou myšleny rozsahy, které tvoří více referencí než počáteční a konečná. Příkladem komplexního rozsahu může být vzorec `=SUM(B2:D3:C4)`. Takto zapsaný vzorec vyhodnotí komplexní rozsah jako rozsah dvou adres buněk. Přičemž první adresu tvoří adresa nejmenšího indexu řádku a sloupce ze specifikovaného komplexního rozsahu, a tu druhou zase největší index řádku a sloupce téhož rozsahu. V tomto konkrétním příkladě je vzorec ekvivalentní vzorci `=SUM(B2:D4)`.

5.2.1 Tokeny

Přehled tokenů

Název tokenu	Popis	Výraz
BOOL	booleovský výraz	'TRUE'|'FALSE'
ERROR	chyba	'#NULL!'| '#DIV/0!'| '#VALUE!'| '#NAME?'| '#NUM!'| '#N/A'
ERROR_REF	chybná reference	'#REF!'
INTEGER	celé číslo pro řádkovou referenci	[0-9]+
LOCK_INTEGER	celé číslo s referenčním zámkem	'$' INTEGER
SHORT_USTRING	velká písmena pro sloupcovou referenci	[A-Z]{1-3}
LOCK_SUSTRING	velká písmena s referenčním zámkem	'$' SHORT_USTRING
CELL_REF	referenční adresa buňky	(SHORT_USTRING | LOCK_SUSTRING) (INTEGER | LOCK_INTEGER)
NUMBER	číselná hodnota	INTEGER+ ('.' INTEGER+)? (EXPONENT ('+' | '-')? INTEGER+)?
UDF	funkce definovaná uživatelem	(LETTER | '_' | ELETTER) UDF_CHAR*
NAMED_RANGE	definovaný název	(LETTER | '\' | '_' | ELETTER) NAMED_RANGE_CHAR*
LINK	cíl externí reference	(''' FILE? QUOTED_SHEET_CHAR+ '!')|(FILE? SHEET_CHAR+ '!')(FILE '!')
STRING	textová hodnota	'"' (~('"' | '\') | ('\' .))* '"'
INNER_SR	vnitřní pole strukturované reference	'[' QUOTED_SHEET_CHAR+']'
Zástupné výrazy		
EXPONENT	exponent matematického zápisu čísel	'e'|'E'
FILE	index externího sešitu	'['INTEGER']'
LETTER	znaky malé nebo velké abecedy	[a-zA-Z]
ELETTER	rozšířené znaky	[\u0080-\u009F\u00A1-\uFEFE\uFF00-\uFFFE]
UDF_CHAR	povolené znaky názvů funkcí	LETTER | INTEGER | ELETTER | [_.]
NAMED_RANGE_CHAR	povolené znaky definovaných názvů	UDF_CHAR | '?'
SHEET_CHAR	povolené znaky názvů listů	~['*[]\:/?!();#=<>%&+-^,_"]
QUOTED_SHEET_CHAR	povolené znaky ohraničených názvů listů	~['*[]\:/]

5.2.2 Gramatika parseru

```
formula: expr EOF
expr: compexpr
compexpr:concexpr (compoper concexpr)?
compoper: OP_NE | OP_LE | OP_GE | OP_GT | OP_LT | OP_EQ
concexpr: addexpr (OP_CCT addexpr)*
addexpr: multexpr (addoper multexpr)*
addoper: OP_ADD | OP_SUB
multexpr: powexpr (multoper powexpr)*
multoper: OP_MUL | OP_DIV
powexpr: unaryexpr (OP_POW unaryexpr)?
unaryexpr: addoper unaryexpr | atom
atom: L_PAR expr R_PAR | exactatom | function | reference
exactatom: number | string | bool | error
reference: refintersection
refintersection: refexternal (OP_INT refexternal)?
refexternal: externalfile refrange | refrange
externalfile: LINK
refrange: rowrange | colrange | cellrange | reflit
reflit: singlecell | namedrange
cellrange: singlecell OP_RNG singlecell
rowrange: row OP_RNG rows
colrange: col OP_RNG col
row: SHORTUSTRING | LOCK_SUSTRING
col: INTEGER | LOCK_INTEGER
namedrange: definedname namedcolumns?
namedcolumns: innerdefname
    | L_BRC innerrows? ( innerdefname | innerrange)? R_BRC
innerrows: (innerdefname OP_UNI)+
innerrange: innerdefname OP_RNG innerdefname
definedname: NAMED_RANGE | UDF | SHORTUSTRING
innerdefname: NAMED_RANGE | UDF | SHORTUSTRING | INNER_SR
singlecell: CELL_REF
function: functioncall
functioncall: functionname L_PAR ( expr (OP_UNI expr)* )? R_PAR
functionname: functionnamelit
functionnamelit: XLLPREFIX? UDF | BOOL | SHORTUSTRING
bool: BOOL
error: ERROR | ERROR_REF
number: NUMBER | INTEGER
string: STRING
```

5.3 Gramatika parseru jazyka Power Query M

Pro alespoň základní porozumění mashup dotazům jazyka Power Query M bylo nutné navrhnout jejich parser. Podpora všech konstruktů tohoto jazyka je mimo rozsah práce, proto byly podporované konstrukty parseru omezeny na základní dedukci technologie a druh zdroje externích dat.

Testovacími příklady definic dotazů byly ručně vytvořené Excelovské soubory s importovanými tabulkami z databází pomocí grafického průvodce importu dat v Excelu. Konkrétním příkladem dotazu je uveden dotaz na tabulku `t2` schématu `sch1` z databáze `manta` na lokálním serveru.

```
1 shared #"sch1_t2" = let
2     Zdroj = Sql.Databases("192.168.0.1"),
3     manta = Zdroj{[Name="manta"]}[Data],
4     sch1_t2 = manta{[Schema="sch1",Item="t2"]}[Data]
5 in
6     sch1_t2
```

Druhým příkladem je dotaz na stejnou tabulku s využitím ODBC ovladače v Power Query dotazu. Hodnota tabulky, která je přiřazená proměnné `Zdroj`, má jinou strukturu. Z toho důvodu má výběr jejích záznamů jiný tvar než v prním příkladě, kdy byla použita vestavěná funkce pro nativní získání tabulky databází Microsoft SQL Server.

```
1 shared t2 = let
2     Zdroj = Odbc.DataSource("dsn=Manta", [HierarchicalNavigation=true]),
3     manta_Database = Zdroj{[Name="manta",Kind="Database"]}[Data],
4     sch1_Schema = manta_Database{[Name="sch1",Kind="Schema"]}[Data],
5     t2_Table = sch1_Schema{[Name="t2",Kind="Table"]}[Data]
6 in
7     t2_Table
```

Z uvedených příkladů lze vidět, že pro nejzákladnější dotazy je nutné rozpoznávat hlavně funkce a operátory výběru z proměnné.

Pro zpracování funkce lze na základě jejího názvu pomocí předkonfigurované mapy hodnot dedukovat technologii a druh navráceného objektu. Funkce mohou obsahovat buď povinné argumenty, které jsou psány do závorky a jsou oddělené čárkami, nebo nepovinné argumenty, jejichž hodnoty spolu s jejich názvy jsou psané do hranatých závorek.

Druhým z důležitých podporovaných konstruktů je operátor přístupu k položce záznamu jak pomocí jmenných identifikátorů, tak pomocí číselného indexu. Pomocí získání proměnné, ze které jsou položky vybírány, lze zjistit typ a technologii zdrojové proměnné, čímž lze odvodit typ a technologii vybíráné položky.

Vynechanými konstrukty jsou transformační nebo tabulkové funkce a aritmetické operace, jelikož jazyk je příliš komplexní pro sestavení prototypu.

5.3.1 Tokeny

Přehled tokenů	
WORD	kvalifikované jméno
QPAR	slovo v uvozovkách
CWORD	komplexní kvalifikované jméno
FUNCTION_NAME	název funkce
Zástupné výrazy	
NUMBER	číselné hodnoty
LETTER	znaky malé nebo velké abecedy
ELETTER	rozšířené znaky

```
(LETTER | ELETTER | '_' | NUMBER)+
'"' ( ~('"' | '\') | ('\' .))* '"'
'#' QPAR
WORD '.' WORD
```

```
[0-9]
[a-zA-Z]
[\u0080-\u009F\u00A1-\uFEFE\uFF00-\uFFFE]
```

5.3.2 Gramatika parseru

```
section: exprs EOF;

exprs: assign_expr (OP_EOL assign_expr)*;

assign_expr: variable OP_EQ expr

expr: function
    | indexing
    | argument
    | type_decls
    | indexing_argument;

function: FUNCTION_NAME L_PAR function_arguments  R_PAR

function_arguments: ( expr (OP_EOL expr)*)?;

indexing: variable indexing_argument L_BRK variable R_BRK

indexing_argument: L_BRC (argument (OP_EOL argument)*) R_BRC;

argument: value
    | L_BRK (values (OP_EOL values)*) R_BRK
    | variable

type_decls: L_BRC type_decl (OP_EOL type_decl)* R_BRC

type_decl:  L_BRC value OP_EOL type R_BRC

values: variable OP_EQ ( value | variable)

value: QPAR;

variable: WORD
    | CWORD
    | FUNCTION_NAME;
```

Kapitola 6

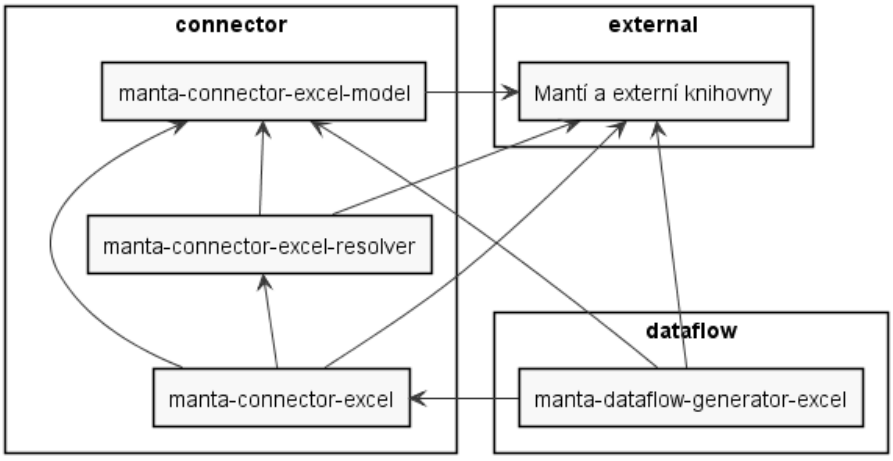
Implementace

V této kapitole bude popsána implementace prototypu rozšíření Manta Flow analyzující Excelovské sešity. Zmíněny budou zajímavé nebo náročné části celé implementace. Představena bude také knihovna Apache POI umožňující přistupovat k objektům Excelovských sešitů.

Prototyp rozšíření se skládá ze dvou modulů. Prvním modulem, který bude představen je `Connector`, který má na starosti načítání vstupního souboru do hierarického stromu, jehož struktura je definována datovým modelem.

Druhým z představených modulů je modul `Dataflow generator`, pomocí kterého jsou vyhledávány datové toky mezi objekty, čímž je vytvořen podgraf datových toků, který je později nahrán do grafové databáze serveru.

Závislosti mezi těmito moduly a jejich balíčky jsou vyobrazeny na diagramu 6.1.



Obrázek 6.1: Diagram závislosti modulů

6.1 Knihovna Apache POI

Apache POI je open-source knihovna vyvíjená organizací Apache Software Foundation pro programovací jazyk Java. Apache POI slouží k manipulování se soubory dílčích nástrojů Microsoft Office ve formátu Office Open XML (OOXML) a starších Microsoft OLE 2 Compound Document (OLE2). Jedním z přispěvovatelů k vývoji byla firma SourceSense v partnerství s firmou Microsoft. [40]

Samotný název knihovny je odvozen od POIFS (zkratka Poor Obfuscation Implementation File System), čímž je interně označován formát OLE2. Jednotlivé třídy Excelovských objektů knihovny mají prefixy podle formátu souborů, se kterými pracují. Pro novější OOXML je tento prefix XSSF (XML Spreadsheet Format), kdežto starší formát OLE2 má prefix HSSF (Horrible Spreadsheet Format). [13] [41]

Kromě Excelovských formátů HSSF a XSSF podporuje knihovna mimo jiné i následující formáty:

- HWPF - binární formát dokumentu Microsoft Word
- XWPF - XML formát dokumentu Microsoft Word
- HSLF - binární formát prezentací Microsoft PowerPoint
- XSLF - XML formát prezentací Microsoft PowerPoint
- HDGF - binární formát diagramů Microsoft Visio
- XDGF - XML formát diagramů Microsoft Visio
- HPBF - binární formát editoru Microsoft Publisher
- HSMF - binární formát souboru Microsoft Outlook

Podpora nových Excelovských OOXML formátů v podobě komponent XSSF byla přidána ve verzi 3.5. Od verze 3.8 beta3 jsou taky dostupné třídy SXSSF (Streaming XML Spreadsheet Format), kterými je řešena paměťová náročnost standardního XSSF. SXSSF využívá limitování přístupu k buňkám, které se nacházející mimo určité pohyblivé okno buněk v paměti. V rámci prototypu tato možnost zůstala neprozkoumána, avšak při otevírání extrémně rozsáhlých sešitů je paměťová limitace XSSF zřejmá.

Pomocí Apache POI je možné přistupovat k listům, jejich buňkám a nativním Excelovským tabulkám. Ostatní elementy datového modelu nemají metody zpřístupňující všechny potřebné informace, a proto je nutné je resolvovat pomocí parsování vnitřních xml souborů za použití vztahů pro přistupování k potřebným zdrojovým xml souborům.

6.2 Connector

Modul `Connector` se skládá ze tří balíčků:

- `manta-connector-excel-model` - obsahuje rozhraní datového modelu Excelu. Neobsahuje definici logiky.
- `manta-connector-excel-resolver` - obsahuje implementaci rozhraní datového modelu a konstruktory objektů, které vytváří instance z vstupního souboru.
- `manta-connector-excel` - obsahuje logiku načítání souboru, volá resolving kořenového objektu hierarchie modelu.

Z těchto balíčků obsahuje největší množství vlastního kódu balíček resolver, proto bude dále popsána pouze jeho funkčnost.

Pojmem resolving se míní proces, kdy z konkrétního Excelovského souboru jsou získávány informace, podle kterých je pak sestavována hierarchická struktura konkrétních instancí objektů, které implementují rozhraní datového modelu. Při resolvingu jednotlivých objektů mohly nastat tři případy.

Prvním případem je, že daný objekt a jeho informace jsou dostatečně reprezentovány pomocí tříd knihovny Apache POI. V tomto případě je resolving poměrně jednoduchý. Příkladem tohoto případu mohou být nativní Excelovské tabulky, listy a sešity.

Druhou možností je objekt, který je dobře popsán ve vlastním definičním XML souboru uvnitř Excelovského souboru. V tomto případě je pomocí Apache POI zpřístupněn jeho definiční XML soubor, který je potom pomocí knihovny dom4j zparsován a potřebné informace z něj jsou získány jako uzly XML hierarchie jazykem XPath. Tímto způsobem je řešen resolving kontingenčních a dotazových tabulek.

Posledním případem je, že daný objekt vlastní definiční soubor nemá, a proto je pro jeho resolving nutné navrhnout způsob, pomocí kterého lze objekt najít a reprezentovat v hierarchii datového modelu. Tento případ nastává pouze u statických tabulek Excelovských listů.

6.2.1 Vyhledávání statických tabulek

Vyhledávání statických tabulek je součástí resolvingu listů sešitu a probíhá až po přidání Excelovských a kontingenčních tabulek daného listu. Algoritmus vyhledávání statických tabulek je rozdělen do tří fází.

První fází je hledání obdélníkových oblastí vyplněných buněk v listu sešitu. Tato fáze využívá cyklu přes všechny vyplněné buňky poskytnuté řádkovými a buňkovými iterátory listů knihovny Apache POI. Pro každou buňku se zkontroluje příslušnost k již existujícím objektům typu `ICellTable` (statické, nativní i kontingenční tabulky).

Příslušnost určité buňky k existující tabulce je zkontrolována pomocí metody `isAdjacent(CellAddress ca)`, která je definovaná v abstraktní třídě `AbstractCellTable`. Při zjištění příslušnosti jsou rozměry tabulky rozšířeny tak, aby pojmuly novou buňku. V opačném případě je vytvořena nová statická tabulka o rozměru jedné buňky a vložena do seznamu elementů listu.

```
/*
 * @param ca cell address
 * @return true if cell address is adjacent to this table.
 */
public boolean isAdjacent(CellAddress ca) {
    return isInside(
        // the cell is inside the table
        ca.getRow() - 1, ca.getColumn())
        && beginning.getColumn() == ca.getColumn()

        // the cell is one field to the right of the beginning of a table
        || isInside(ca.getRow(), ca.getColumn() - 1)
        && beginning.getRow() == ca.getRow()

        // the cell is one field to the right
        // and one field under the beginning of a table
        || isInside(ca.getRow() - 1, ca.getColumn())
        && isInside(ca.getRow(), ca.getColumn() - 1)

        // the cell is one field to the left
        // and one field under the beginning of a table
        // (occurs when the first header of a table is blank (usually ID))
        || isInside(ca.getRow() - 1, ca.getColumn() + 1)
        && beginning.getRow() == ca.getRow() - 1
        && beginning.getColumn() == ca.getColumn() + 1

        // the cell is inside the table
        || isInside(ca.getRow(), ca.getColumn());
    }
}
```

Druhou fází vyhledávání statických tabulek je dedukce orientace polí dané tabulky pomocí metody `setTableOrientation(XSSFSheet sheet)`. Tato dedukce probíhá v cyklu o počtu opakování, který je rovný většímu z rozměrů tabulky, ale ne většímu než přednastavená konstanta. V každém cyklu je zkontrolováno, jestli všechny buňky v `i`-tém řádku a `i`-tém sloupci obsahují stejný datový typ, případně stejný vzorec. Vzorce je před porovnáním nutné zbavit referencí na buňky, jelikož se budou lišit s ohledem na pozici dané buňky. Pokud v je `n`-tém cyklu zaznamenán nehomogenní řádek či sloupec, cyklus je přerušen a orientace záznamů je nastavena.

V případě, že první řádek či sloupec jsou homogenní a jejich datový typ je

textový řetězec, je nastaven příznak `portraitHeaders` či `landscapeHeaders` respektivně na `true`. Tímto je po přerušení cyklu dedukováno, zda-li první záznam obsahuje názvy polí či tabulka záhlaví neobsahuje vůbec.

```
/*
* @param sheet sheet where table is located
*/
public void setTableOrientation(XSSFSheet sheet) {
    boolean exitFlag = false, landscapeHeaders = false, portraitHeaders = false;
    for (int i = 0; i < Math.min(Math.max(getTableHeight(), getTableWidth()),
        MAX_TABLE_FIELDS); i++) {
        OrientationHelper rowHelper = null;
        OrientationHelper colHelper = null;
        // row traversal
        if (i < getTableHeight()) {
            rowHelper = crawlTableRow(i, sheet);
            if (i == 0 && rowHelper.isHomogeneous()
            && rowHelper.getOutputDataType() == CellType.STRING) {
                portraitHeaders = true; // first row can be header
            }
            if (!rowHelper.isHomogeneous()) {
                setPortrait(true); // not homogeneous, is vertical
                exitFlag = true; // end cycle
            }
        }
        // column traversal
        if (i < getTableWidth()) {
            colHelper = crawlTableCol(i, sheet);
            if (i == 0 && colHelper.isHomogeneous()
            && colHelper.getOutputDataType() == CellType.STRING) {
                landscapeHeaders = true; // first col can be header
            }
            if (!colHelper.isHomogeneous()) {
                setPortrait(false); // not homogeneous, is horizontal
                exitFlag = true; // end cycle
            }
        }
        if (exitFlag) {break;}
    } // cycle end
    // set headers if the table is not a single cell
    if (getTableHeight() != 1 && getTableWidth() != 1) {
        if (isPortrait()) {
            setHasHeaders(portraitHeaders);
        } else {
            setHasHeaders(landscapeHeaders);
        }
    }
}
```

Metoda `setTableOrientation(XSSFSheet sheet)` volá uvnitř metodu `crawlTableRow(int rowIndex, XSSFSheet sheet)` pro prozkoumání homogenity v řádku. Pro kontrolu homogenity ve sloupci slouží obdobná metoda `crawlTableCol(int colIndex, XSSFSheet sheet)`. Metody buňky iterují odlišně, proto je potřeba mít oba způsoby ve vlastních metodách.

```
1  /**
2   * Iterate over cells in a row to see if the row is homogeneous.
3   *
4   * @param rowIndex index of the row that is being crawled.
5   * @param sheet source sheet.
6   * @return orientation helper object used in determining table's record orientation.
7   */
8  private OrientationHelper crawlTableRow(int rowIndex, XSSFSheet sheet) {
9      Row row = sheet.getRow(getAbsoluteRowNum(rowIndex));
10     OrientationHelper rowHelper = new OrientationHelper();
11     boolean first = rowIndex == 0;
12     // iterate over cells in a row
13     for (int colIndex = 0; colIndex < getTableWidth(); colIndex++) {
14         Cell cell = row.getCell(getAbsoluteColNum(colIndex),
15         Row.MissingCellPolicy.CREATE_NULL_AS_BLANK);
16         if (!rowHelper.adjust(colIndex, cell, first)) {
17             // row is not homogeneous
18             break;
19         }
20     }
21     return rowHelper;
22 }
```

Pro dedukci orientace je využita pomocná třída `OrientationHelper` udržující informace o datovém typu, návratovém datovém typu vzorce, vzorci a homogenitě daného řádku či sloupce. Třída `OrientationHelper` obsahuje metodu `adjust(int index, Cell cell, boolean first)`, která tyto zmíněné informace o řádku či sloupci porovná a aktualizuje údaje s ohledem na buňku předanou v argumentu.

Třetí fází resolvingu statických tabulek je přidání jednotlivých polí. Pokud neobsahuje záhlaví, je iterováno po indexech řádků či sloupců v závislosti na orientaci tabulky. Z prvních buněk těchto potenciálních polí je odvozen datový typ nebo vzorec pole a pole je vytvořeno a přidáno.

V opačném případě, kdy tabulka obsahuje záhlaví, je v cyklu potenciálních polí získána první buňka jako buňka záhlaví. Buňka, od které je odvozen vzorec či datový typ, je pak získávána v cyklu jako první neprázdné pole, čímž je ošetřena možnost pole s nepovinnými hodnotami. Tato metoda je sdílena s nativními tabulkami, jelikož informace o datovém typu sloupce není součástí definičního souboru tabulky.

```
1  public void addTableFields(XSSFSheet sheet) {
2      // if the table is portrait, the number of fields is equal to the width of the table,
3      // if not it is equal to the height of the table
4      int tableFieldCount = isPortrait() ? getTableWidth() : getTableHeight();
5      // if the table is portrait, the number of records is equal to the height of the table,
6      // if not it is equal to the width of the table
7      int tableFieldLength = isPortrait() ? getTableHeight() : getTableWidth();
8      if (hasHeaders()) { // traverse the fields
9          for (int fieldIndex = 0; fieldIndex < tableFieldCount; fieldIndex++) {
10             // if the table is portrait, the header cell is the first cell in each column,
11             // if not it is the first cell in each row
12             Cell headerCell = isPortrait()
13                 ? sheet.getRow(getAbsoluteRowNum(0))
14                 .getCell(getAbsoluteColNum(fieldIndex))
15                 : sheet.getRow(getAbsoluteRowNum(fieldIndex))
16                 .getCell(getAbsoluteColNum(0));
17             // traverse the records until a valid cell is reached
18             for (int recordIndex = 1; recordIndex < tableFieldLength;
19             recordIndex++) {
20                 // the value cell is the next cell after the header
21                 Cell valueCell = isPortrait()
22                     ? sheet.getRow(getAbsoluteRowNum(recordIndex))
23                     .getCell(getAbsoluteColNum(fieldIndex))
24                     : sheet.getRow(getAbsoluteRowNum(fieldIndex))
25                     .getCell(getAbsoluteColNum(recordIndex));
26                 // there might be blank or null cells if the field is optional
27                 if (valueCell != null) {
28                     addTableField(headerCell, valueCell, fieldIndex);
29                     break;
30                 }
31                 // if the end was reached without finding any non blank cells
32                 if (recordIndex == tableFieldLength - 1) {
33                     addTableField(headerCell, null, fieldIndex);
34                 }
35             }
36         }
37     } else { // table doesn't have any headers.
38         for (int fieldIndex = 0; fieldIndex < tableFieldCount; fieldIndex++) {
39             if (isPortrait()) {
40                 addTableField(sheet.getRow(getAbsoluteRowNum(0))
41                 .getCell(getAbsoluteColNum(fieldIndex)), fieldIndex);
42             } else {
43                 addTableField(sheet.getRow(getAbsoluteRowNum(fieldIndex))
44                 .getCell(getAbsoluteColNum(0)), fieldIndex);
45             }
46         }
47     }
48 }
```

6.2.2 Získávání kompletní informace o externím dotazu

Resolving externích spojení sešitu může nabývat dvou podob. První možností je, že dotaz je vytvořen nástrojem Microsoft Query, čímž je jeho zpracování jednoduché. Takový dotaz se skládá pouze ze dvou atributů záznamu v souboru `/xl/connections.xml`. Prvním je `connectionString`, který popisuje způsob připojení k externímu zdroji a druhým je `command` obsahující dotaz v dotazovacím jazyce daného zdroje.

Druhou možností je využití dotazovacího jazyka Power Query M, který je v Excelu jako rozšíření od verze 2010. Informace o dotazech Power Query jsou udržovány v souboru vlastního XML, kde jsou uchovávány zakódovné v base64 řetězci znaků. Část infromace o spojení Microsoft Query v souboru `/xl/connections.xml` zůstává a `connectionString` obsahuje informace ukazující do sešitu na mashup dotaz a `command` obsahuje `SELECT` příkaz pro vybrání všech sloupců výstupní proměnné dotazu definovaného v Power Query M jazyce.

Obě tyto možnosti mohou využít datového modelu Power Pivot, čímž se skryje přímý ukazatel na spojení. Pro získání skryté informace je nutné dekomprimovat bytový stream souboru datového modelu `/xl/model/item1.data`. Tento soubor používá komprimační algoritmus Xpress Compression Algorithm (XCA).

Postup resolvingu externího spojení v implementaci je následující:

- načíst `/xl/connections.xml` a vytvořit instance spojení
- načíst a dekomprimovat `/xl/model/item1.data`, pomocí něhož obohatit vytvořené instance spojení o příkaz, pokud byl daný dotaz uložen do datového modelu Power Pivot
- načíst `/customXml/item1.xml` a obohatit instance o Power Query dotazy pokud používají dotazy definované v Power Query M

Resolving souboru `/xl/connections.xml` je poměrně jednoduchý, jelikož jsou jeho data v formátu XML, čímž je snadno čitelný bez nutnosti dekomprimace či jakéhokoliv dalšího zpracovávání. Pro resolving tohoto souboru je použita statická metoda `addConnection()` třídy `DBConnectionHelper`. Výsledkem resolvingu této metody je vytvoření a přidání instancí `DBConnection` instanci objektu `IExcelWorkbook`.

Zpracování souboru datového modelu `/xl/model/item1.data` se skládá z celkem čtyř kroků. Nejprve je nutné si z celého vstupního proudu tohoto souboru načíst hlavičku. Z hlavičky lze zjistit pozici a délku elementu `Virtual Directory` ve vstupním proudu.

Druhým krokem je zpracování tohoto elementu. `Virtual Directory` obsahuje pozice a délky dílčích souborů zakódovaných ve vstupním proudu. Jednotlivé záznamy v tomto elementu však neobsahují název, ale pouze iden-

tifikátor. Jediným souborem obsahující název je soubor `LOG`, který obsahuje element `BackupLog`.

Třetím krokem je zpracování `BackupLog`. Element `BackupLog` obsahuje podrobnější informace o souborech nacházejících se v tomto proudu jako je název, typ, skupina, dekomprimovaná velikost a identifikátor, spojující záznamy v `BackupLog` se záznamy `Virtual Directory`.

Posledním krokem je získání souboru `Sandbox` pomocí kombinování informací z `Virtual Directory` a `BackupLog`. Díky těmto informacím je možné získat pole komprimovaných bytů tohoto souboru. Soubory datového modelu Power Pivot používají algoritmus `MS-XCA`. Pro dekomprimaci byl implementován dekomprimační algoritmus podle dokumentace ke specifikaci datového modelu `MS-XLDM`. Výsledkem dekomprimace je XML soubor, ze kterého lze získat požadovaný příkaz dotazu, který je přidán odpovídající instanci objektu `DBConnection`.

Poslední fází resolvingu externích spojení je získání definic dotazů Power Query M. Pro získání souboru s definicemi dotazů je nutné lokalizovat soubor uvnitř složky `/customXml`, který podléhá specifikaci `MS-QDEFF`. Z tohoto souboru je nutné získat řetězec znaků, který je pak dekódován pomocí base64. Výsledkem dekódování je proud bytů, ze kterých jsou získány čtyři soubory, jejichž byty jsou předcházeny čtyřmi byty, které obsahují délku bytů daného souboru. Prvním z těchto čtyř souborů je soubor Package Parts, který je OPC balík obsahující soubor `/Formulas/Section1.m`, který obsahuje všechny Power Query dotazy definované v tomto sešitě. Z tohoto souboru jsou jednotlivé dotazy získány a přiřazeny odpovídající instanci objektu `DBConnection`.

6.3 Dataflow

Modul `Dataflow` obsahuje na nejvyšší úrovni pouze jeden balíček, a to balíček `manta-dataflow-generator-excel`. Tento balíček je dále členěn následovně:

- `analyzers` - obsahuje analyzátory jednotlivých objektů hierarchie datového modelu.
- `connection` - obsahuje pomocné třídy propojující uzly s uzly externího spojení nebo souboru.
- `el` - obsahuje pomocné třídy pro parsování Excelovských vzorců.
- `helper` - obsahuje pomocné třídy pro manipulaci s výsledným grafem datových toků.
- `m` - obsahuje pomocné třídy pro parsování Power Query M dotazů.

Při procesu generování dataflow se z hierarchické struktury odpovídající definovaného data modelu vytvoří uzly grafu, které jsou poté spojeny hranami, které odpovídají toku dat mezi nimi. Jednotlivé reference uzlů objektů

uvnitř souboru jsou zapsané buď pomocí Excelovských vzorců, nebo pomocí textových či číselných identifikátorů, tak jak jsou definovány v jejich XML souborech.

Pro zpřístupnění cílových uzlů externích dotazů je použita společná třída pro dataflow generátory `DataFlowQueryService`. Tato třída obsahuje metodu `getDataFlow()`, která po předání textového řetězce obsahující dotaz a objektu typu `Connection` obsahující informace o datovém spojení, vrátí množinu cílových uzlů existujících v instanci grafové databáze.

Jako ukázka implementace bude představen způsob, jakým jsou zpřístupňovány referované objekty pomocí Excelovských adresových referencí při parsování Excelovských výrazů.

6.3.1 Zpřístupnění objektu pomocí Excelovských výrazů

Při parsování Excelovských výrazů pomocí ANTLR se sestrojí abstraktní syntaktický strom (zkrácně AST), jehož uzly jsou od kořene hlouběji zpracovávány. Důležitými uzly stromu pro analýzu datových toků jsou funkce, externí reference, referenční operace a samotné reference.

Metoda `resolveExcelObject()` vrací pole odkazované Excelovskou referencí. Jejími argumenty je jak uzel AST, který obsahuje typ a obsah parsované reference, tak textový řetězec obsahující externí část reference, pokud je referencován objekt z jiného listu nebo sešitu.

Nejprve, pokud argumenty volání externí část reference obsahují, se vyhodnotí externí výraz metodou `resolveExternalExcelObject()`. Tato metoda vrací objekt typu `ExcelNode`, konkrétně pak nejčastěji objekty typu `IExcelSheet` či typu `IExcelWorkbook`, pokud v externí části reference není specifikován konkrétní list. Metoda pomocí regulárního výrazu nejprve externí výraz rozloží na část externího sešitu a listu. Pokud je část externího sešitu specifikovaná, nejprve se dohledá pomocí indexu daný externí sešit. Z tohoto sešitu se pak, pokud je specifikovaný i název listu, dohledá i list daného sešitu. Tímto je získán kontext pro vyhledávání cílu reference. Pokud je externí reference prázdná, použije se sešit a list objektu, který referencí odkazuje.

V další části se logika odvíjí od typu reference. Jednotlivými typy referencí mohou být:

- reference na adresu buňky
- reference na rozsah adres buněk
- strukturovaná reference (definovaný název nebo tabulka)

V případě, že referencí je adresa jedné buňky, je logika jednoduchá. Nejprve se z externího listu zjistí podle adresy, jaké tabulce buňka patří, a poté je pomocí adresy buňky získáno konkrétní pole tabulky, ve kterém se buňka nachází.

Obdobně je zpracovávána reference typu rozsahu adres buněk. V případě, že jde o rozsah sloupců nebo řádků, je nutné adresy počátku a konce normalizovat tak, aby obsahovaly obě dimenze reference. Poté vše probíhá stejně jako by šlo o běžnou buněčnou referenci.

Pokud je reference strukturovaná, čímž je myšleno, že reference neobsahuje adresu buněk, ale nějaký platný identifikátor, provede se nejdříve pokus o vyhledání daného identifikátoru mezi definovanými názvy. Pokud cílový sešit takto pojmenovaný definovaný název obsahuje, je vrácen. V případě, že žádný takto pojmenovaný definovaný název v sešitu neexistuje, se předpokládá, že tento identifikátor náleží tabulce, která je dále hledána nejprve ve specifikovaném listu, případně v celém specifikovaném sešitě.

Když je tabulka dohledána, jsou dále zpracovávány identifikátory sloupců. Jednotlivé sloupce mohou být referovány jednak po jednom za použití jejich názvu, jednak jako rozsah použitím operátoru rozsahu a názvů počátečního a konečného sloupce. Po výsledném zpracování je vrácen seznam referencovaných sloupců dané tabulky.

6.4 Testování

V této sekci bude popsané testování implementovaného prototypu. Testování se týká modulů Connector a DataFlow. Pro testování byly využity jak soubory uměle vytvořené pro otestování určité funkčnosti, tak reálné soubory získané od klientů společnosti Manta.

6.4.1 Connector testy

Pro testování funkčnosti Connectoru je implementována parametrizovaná testovací třída `LoadManagerTest`. Uvnitř této třídy je pomocí testovací metody `executeLoadTest()` načtena množina Excelovských sešitů v testovaném adresáři do instance třídy `ExcelWorkbookManager`. Po načtení všech dílčích Excelovských sešitů je získán řetězec znaků popisující celou strukturu sešitů adresáře tak, jak je definovaná v datovém modelu.

Získaný řetězec znaků je pak porovnán s očekávaným řetězcem znaků, který je načten z textového souboru očekávné hodnoty testovaného adresáře. Správnost testu spočívá v platné instanci této třídy a v porovnání skutečné a očekávané hodnoty řetězce struktury.

6.4.2 Dataflow generator testy

Parametrizovaná testovací třída `ParserTest` slouží k testování správnosti parseru Excelovských vzorců. Testovací příklady jsou načítány ze souborů a pomocí parseru vygenerovaném gramatikou nástroje ANTLR je získán abstraktní syntaktický strom. Správnost testu závisí na platné instanci syntak-

tického stromu a absenci varovných logovacích výpisů. Příklady testovaných vzorců pokrývají podporované konstrukty gramatiky parseru tohoto jazyka.

Obdobně jako třída `ParserTest`, tak třída `ParserSectionTest` testuje správnost parseru výrazů, konkrétně Power Query M dotazů. Testovací příklady byly získány ze skutečných dotazů vytvořených importováním dat v Power Query grafickém rozhraní Excelu.

Parametrizovaná testovací třída `FilesFlowTest` dědí z abstraktní testovací třídy `ExcelTestBase`. `ExcelTestBase` obsahuje pomocnou metodu pro inicializaci databázových slovníků `createDictionary()`. Touto metodou se vytvoří záznam ve slovníku zdrojů, ke kterým se později mohou pojit uzly z Excelovských souborů. Jednotlivé záznamy jsou definované v konfiguračním spring souboru.

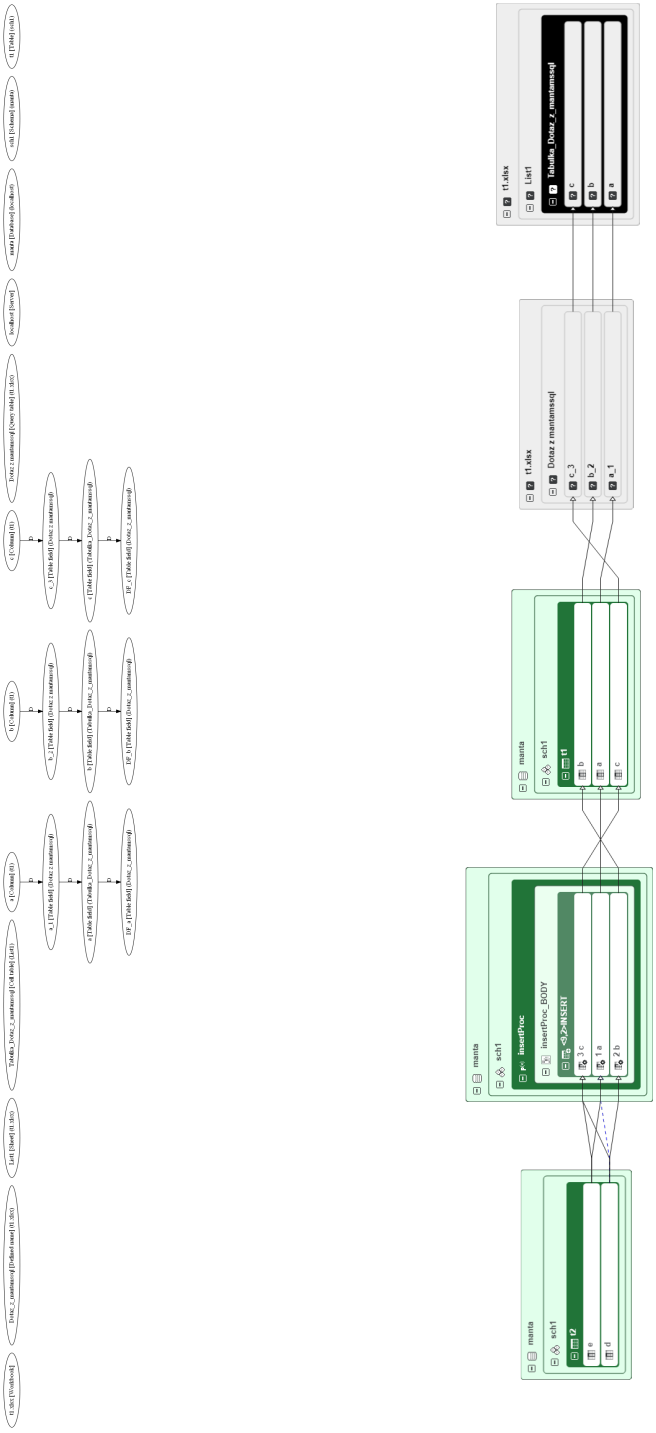
Princip `FilesFlowTest` spočívá v načtení Excelovských sešitů uvnitř adresáře a v jejich analyzování. Výsledkem analýzy je graf datových toků, který je uložen do souboru a je porovnán s očekávanou podobou grafu. Pokud je při spuštění testu nastavený parametr výpisu obrázku, je výsledný graf exportován ve formátu `png` do souboru pomocí nástroje `GraphViz`.

6.4.3 Testovací soubory

V této sekci budou představeny jednotlivé testovací adresáře spolu s jejich soubory, které jsou přítomné na přiloženém CD. Tyto soubory jsou uměle vytvořené tak, aby pokryly důležité funkčnosti prototypu. Testovací soubory jsou sdílené mezi testy modulu `Connector` a modulu `Dataflow generator`.

```
filesFlowTest
├── complexTest ..... komplexní test všech funkcí
├── connectionTest ..... test importu z externích zdrojů
├── definedNameTest ..... test definovaných názvů
├── externalSheetReferenceTest ..... test referencí napříč soubory
├── fileTest ..... test importu dat ze souboru
├── chartTest ..... test grafů a kontingenčních grafů
├── nativeTableTest ..... test Excelovských tabulek
├── pivotTableTest ..... test kontingenčních tabulek
├── slicerTest ..... test průřezů
└── staticTableTest ..... test statických tabulek
```

Obrázek 6.2: Adresář testovacích souborů



(a) Vizualizace nástrojem GraphViz (b) Vizualizace nástrojem Manta Flow

Obrázek 6.3: Graf importu tabulky z Microsoft SQL Server do Excelu

Závěr

Cílem této práce bylo analyzovat způsob ukládání Excelovských sešitů a vytvořit prototyp implementace analyzátoru datových toků těchto sešitů. V teoretické části byl představen nástroj Microsoft Excel a společnost Manta s jejím nástrojem Manta Flow. Následně byly popsány cíle implementace prototypu spolu s technologiemi použitými k jeho realizaci. Značná část práce byla věnována analýze stuktury Excelovských souborů, podle níž byl navrhnut datový model, který byl v následující kapitole popsán společně s gramatikou parserů jazyků použitých v Excelu. V poslední kapitole byla popsána implementace spolu s vybranými řešenými problémy a testováním.

Známé nedostatky prototypu se týkají především resolvingu statických tabulek, jelikož jsou tyto objekty v praxi často využívané díky jednoduchosti jejich použití. Prototyp dokáže vyhledat pouze jasně definované tabulky s homogenními poli. Jakékoliv použití sjednocených buněk, záhlaví sloupců o více buňkách, řádků s mezisoučty či vynechání mezer za účelem stylování znemožní algoritmu správně posoudit existenci a vlastnosti tabulky. Tyto objekty však nejsou přímo spojené s externím zdrojem, proto nejsou pro analýzu datových toků příliš významné.

Možnosti dalšího rozšíření prototypu spočívají v podpoře více formátů Excelovských sešitů. Nejsnazším kandidátním formátem na další podporu je `xlsb`, jelikož jeho struktura je stejná jako standardní OOXML sešit, avšak jeho data jsou uložena binárně, namísto XML. Obtížnější, avšak více žádané by bylo rozšířit podporu o staré proprietární binární formáty s koncovkou `xls`. Tyto formáty mají výrazně odlišnou strukturu, avšak výskyt reportů tohoto formátu je v BI prostředí častější.

Dalším možným rozšířením by bylo zlepšení podpory dotazovacího mashup jazyka Power Query M. V současné době dokáže prototyp odhalit pouze původ z jediného zdroje. V reálné praxi však tento jazyk umožňuje kombinovat data z více zdrojů, včetně jejich transformace. Kompletní podpora tohoto jazyka byla mimo rozsah prototypu a implementována byla jen jako ověření koncepce.

Výsledkem diplomové práce bylo naplnění stanovených cílů a zadání práce jako celek považuji za kompletně splněné. Při implementování prototypu jsem se seznámil s programátorskou praxí na větším projektu a získal mnoho cenných zkušeností, které věřím, že v další kariérní praxi zúročím.

Literatura

[1] Tools, M.: Manta Flow Pricing, Features, Reviews & Comparison of Alternatives [online]. *GetApp*, [cit. 2019-04-17]. Dostupné z: https://www.getapp.com/business-intelligence-analytics-software/a/manta-flow/

[2] Manta Flow Architecture [online]. [cit. 2019-04-18]. Dostupné z: https://mantatools.atlassian.net/wiki/spaces/MTKB/pages/70230122/MANTA+Flow+Architecture

[3] Walkenbach, J.: *Excel 2003 Power Programming with VBA [online]*. Wiley, 2011, ISBN 9781118043974, [cit. 2019-04-28]. Dostupné z: https://books.google.cz/books?id=XovWrtKUlTsC

[4] Office Open XML - What is OOXML? [online]. 2012, [cit. 2019-03-15]. Dostupné z: http://officeopenxml.com/

[5] Rozhodnuto. S Czech ICT Alliance jedou do Silicon Valley dva startupy - Manta a realPad – Tyinternety.cz [online]. *Tyinternety.cz*, Feb 2015, [cit. 2019-05-01]. Dostupné z: https://tyinternety.cz/startupy/rozhodnuto-s-ict-alliance-jedou-silicon-valley-dva-startupy-manta-realpad/

[6] MANTA – Automate your data lineage [online]. *MANTA*, May 2019, [cit. 2019-04-18]. Dostupné z: https://getmanta.com/

[7] Tech Summary – MANTA [online]. *MANTA*, Apr 2019, [cit. 2019-05-01]. Dostupné z: https://getmanta.com/scanners-and-integrations/tech-summary/

[8] Binstock, A.: Java's 20 Years Of Innovation [online]. *Forbes*, May 2015, [cit. 2019-04-23]. Dostupné z: https://www.forbes.com/sites/oracle/2015/05/20/javas-20-years-of-innovation/

[9] Parr, T.: *The Definitive ANTLR 4 Reference [online]*. Oreilly and Associate Series, Pragmatic Bookshelf, 2012, ISBN 9781934356999, [cit. 2019-04-23]. Dostupné z: https://books.google.cz/books?id=SBXuLwEACAAJ

[10] Porter, B.; Zyl, J. v.; Lamy, O.: Maven – Welcome to Apache Maven [online]. *Maven – Welcome to Apache Maven*, [cit. 2019-04-23]. Dostupné z: https://maven.apache.org/

[11] Spring: the source for modern java [online]. *Spring*, [cit. 2019-04-23]. Dostupné z: https://spring.io/

[12] Apache Subversion [online]. *Apache Subversion*, [cit. 2019-04-23]. Dostupné z: https://subversion.apache.org/

[13] Oliver, A. C.; Klute, R.; Fisher, D.: Component Overview [online]. *Apache POI*, [cit. 2019-04-23]. Dostupné z: https://poi.apache.org/components/index.html

[14] Flexible XML framework for Java. [online]. *dom4j*, [cit. 2019-04-23]. Dostupné z: https://dom4j.github.io/

[15] Package Structure [online]. [cit. 2019-04-20]. Dostupné z: https://c-rex.net/projects/samples/ooxml/e1/Part1/OOXML_P1_Fundamentals_Package_topic_ID0EXOEK.html

[16] Workbook [online]. [cit. 2019-04-25]. Dostupné z: https://c-rex.net/projects/samples/ooxml/e1/Part4/OOXML_P4_DOCX_Workbook_topic_ID0E1C63.html

[17] Worksheets [online]. [cit. 2019-04-24]. Dostupné z: https://c-rex.net/projects/samples/ooxml/e1/Part4/OOXML_P4_DOCX_Worksheets_topic_ID0E6AM4.html

[18] External Data Connections [online]. [cit. 2019-04-25]. Dostupné z: https://c-rex.net/projects/samples/ooxml/e1/Part4/OOXML_P4_DOCX_External_topic_ID0E4ZUBB.html

[19] Hart, J.: Přehled vlastních částí XML - Visual Studio [online]. *Microsoft Docs*, [cit. 2019-04-28]. Dostupné z: https://docs.microsoft.com/cs-cz/visualstudio/vsto/custom-xml-parts-overview

[20] Openspecs-Office: [MS-QDEFF]: Query Definition File Format [online]. *Microsoft Docs*, [cit. 2019-04-29]. Dostupné z: https://docs.microsoft.com/en-us/openspecs/office_file_formats/ms-qdeff/27b1dd1e-7de8-45d9-9c84-dfcc7a802e37

[21] Openspecs-Office: [MS-XLDM]: Spreadsheet Data Model File Format [online]. *Microsoft Docs*, [cit. 2019-05-02]. Dostupné z: https://docs.microsoft.com/en-us/openspecs/office_file_formats/ms-xldm/8c62e8ce-f605-488d-81e9-4ecdb7686a52

[22] Openspecs-Windows: [MS-XCA]: Xpress Compression Algorithm [online]. *Microsoft Docs*, [cit. 2019-05-02]. Dostupné z: https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-xca/a8b7cb0a-92a6-4187-a23b-5e14273b96f8

[23] definedName (Defined Name) [online]. [cit. 2019-04-25]. Dostupné z: https://c-rex.net/projects/samples/ooxml/e1/Part4/OOXML_P4_DOCX_definedName_topic_ID0E2JB4.html

[24] C-rex: *Tables [online]*. [cit. 2019-04-22]. Dostupné z: https://c-rex.net/projects/samples/ooxml/e1/Part4/OOXML_P4_DOCX_Tables_topic_ID0E3YU5.html

[25] QueryTable Data [online]. [cit. 2019-04-22]. Dostupné z: https://c-rex.net/projects/samples/ooxml/e1/Part4/OOXML_P4_DOCX_QueryTable_topic_ID0EF4RBB.html

[26] Pivot Tables [online]. [cit. 2019-04-22]. Dostupné z: https://c-rex.net/projects/samples/ooxml/e1/Part4/OOXML_P4_DOCX_Pivot_topic_ID0EU2Y6.html

[27] chartSpace (Chart Space) [online]. [cit. 2019-04-28]. Dostupné z: https://c-rex.net/projects/samples/ooxml/e1/Part4/OOXML_P4_DOCX_chartSpace_topic_ID0ERXXPB.html

[28] Průřez/Slicer v Excelu (bez kontingenční tabulky) [online]. *Excel-Town.com - školení, konzultace, návody - Excel, Power BI*, [cit. 2019-05-01]. Dostupné z: https://exceltown.com/navody/filtry-razeni-souhrny-atd/prurezslicer-v-excelu-bez-kontingencni-tabulky/

[29] Use Microsoft Query to retrieve external data [online]. *Excel*, [cit. 2019-04-30]. Dostupné z: https://support.office.com/en-us/article/Use-Microsoft-Query-to-retrieve-external-data-42a2ea18-44d9-40b3-9c38-4c62f252da2e

[30] What's new in Excel 2013 [online]. *Excel*, [cit. 2019-04-27]. Dostupné z: https://support.office.com/en-us/article/what-s-new-in-excel-2013-1cbc42cd-bfaf-43d7-9031-5688ef1392fd

[31] Get & Transform in Excel [online]. *Excel*, [cit. 2019-04-27]. Dostupné z: https://support.office.com/en-us/article/get-transform-in-excel-881c63c6-37c5-4ca2-b616-59e18d75b4de

[32] TREACY, M.: Power Query Version Compatibility and Installation [online]. *My Online Training Hub*, [cit. 2019-04-27]. Dostupné z: https://www.myonlinetraininghub.com/power-query-version-compatibility-and-installation

[33] Sheldon, R.: Power BI Introduction: Power Query M Formula Language in Power BI Desktop - Part 6 [online]. *Simple Talk*, Sep 2018, [cit. 2019-04-27]. Dostupné z: https://www.red-gate.com/simple-talk/sql/bi/power-bi-introduction-power-query-m-formula-language-in-power-bi-desktop-part-6/

[34] Aivaloglou, E.; Hoepelman, D.; Hermans, F.: Parsing Excel formulas: A grammar and its application on 4 large datasets [online]. *Journal of Software: Evolution and Process*, ročník 29, 2017, [cit. 2019-04-30].

[35] Using structured references with Excel tables [online]. *Office Support*, [cit. 2019-04-30]. Dostupné z: https://support.office.com/en-us/article/using-structured-references-with-excel-tables-f5ed2452-2337-4f71-bed3-c8ae6d2b276e

[36] Excel functions (alphabetical) [online]. *Office Support*, [cit. 2019-04-29]. Dostupné z: https://support.office.com/en-ie/article/excel-functions-alphabetical-b3944572-255d-4efb-bb96-c6d90033e188

[37] Openspecs-Office: [MS-XLSX]: Formulas [online]. *[MS-XLSX]: Formulas — Microsoft Docs*, [cit. 2019-04-30]. Dostupné z: https://docs.microsoft.com/en-us/openspecs/office_standards/ms-xlsx/3d025add-118d-4413-9856-ab65712ec1b0

[38] Jin, A.: Introduction to Power Query (informally known as "M") Formula Language) [online]. *Microsoft Docs*, [cit. 2019-04-27]. Dostupné z: https://docs.microsoft.com/en-us/previous-versions/mt270235(v=msdn.10)

[39] Hoepelman, D.: XLParser [online]. https://github.com/spreadsheetlab/XLParser, [cit. 2019-05-01].

[40] Microsoft and Sourcesense Partner to Contribute to Open Source, Apache POI to Support Ecma Office Open XML File Formats [online]. *Stories*, Mar 2008, [cit. 2019-04-25]. Dostupné z: https://news.microsoft.com/2008/03/25/microsoft-and-sourcesense-partner-to-contribute-to-open-source-apache-poi-to-support-ecma-office-open-xml-file-formats/

[41] Sundaram, E.: Excelling in Excel with Java [online]. *JavaWorld*, Mar 2004, [cit. 2019-04-28]. Dostupné z: https://www.javaworld.com/article/2072333/excelling-in-excel-with-java.html

Příloha **A**

Seznam použitých zkratek

ANTLR ANother Tool for Language Recognition

AST Abstract Syntax Tree

BI Business Intelligence

CLI Command Line Interface

DrawingML Drawing Markup Language

ECMA European Computer Manufacturers Association

HDGF Horrible Diagram Format

HPBF Horrible Publisher Format

HSLF Horrible Slide Layout Format

HSSF Horrible Spreadsheet Format

HSMF Horrible Stupid Mail Format

HWPF Horrible Word Processor Format

IBM International Business Machines

ICT Information and Communication Technologies

IEC International Electrotechnical Commission

IP Internet Protocol

ISO International Organization for Standardization

MS DOS Microsoft Disk Operating System

MS-QDEFF Query Definition File Format

MS-XLDM Spreadsheet Data Model File Format

ODBC Open Database Connectivity

OLE Object Linking and Embedding

OLE2 OLE 2 Compound Document

OOXML Office Open XML

OPC Open Packaging Conventions

POI Poor Obfuscation Implementation

POIFS Poor Obfuscation Implementation File System

SpreadsheetML Spreadsheet Markup Language

SQL Structured Query Language

SSAS SQL Server Analysis Services

SXSSF Streaming XMLSpreadsheet Format

TCP Transmission Control Protocol

UDF User defined function

XCA Xpress Compression Algorithm

XML Extensible Markup Language

XDGF XML Diagram Format

XSLF XML Slide Layout Format

XSSF XML Spreadsheet Format

XWPF XML Word Processor Format

Příloha **B**

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `src`
  - `impl` ..... zdrojové kódy implementace
    - `Connector` ..... zdrojové kódy modulu Connector
    - `Dataflow` ..... zdrojové kódy modulu Dataflow
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `DP_Micek_Daniel_2019.pdf` ..... text práce ve formátu PDF