FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

ZADÁNÍ DIPLOMOVÉ PRÁCE

Název:	Analýza datových toků v reportovacích nástrojích
Student:	Bc. Petr Košvanec
Vedoucí:	Ing. Michal Valenta, Ph.D.
Studijní program:	Informatika
Studijní obor:	Webové a softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce zimního semestru 2019/20

Pokyny pro vypracování

1. Formou rešerše proveďte analýzu datového modelu a datových toků v těchto reportovacích nástrojích: OBIEE od Oracle, Cognos od IBM, SSRS a Excel od Microsoft.
2. Na základě rešerše navrhněte sjednocený datový model pro analýzu datových toků v těchto nástrojích.
3. Implementujte prototyp modulu pro nástroj Manta, který analyzuje datové toky v nástroji Cognos a provede jejich transformaci do sjednoceného formátu navrženého v bodě 2. Prototyp řádně otestujte a zdokumentujte.

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

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

V Praze dne 18. září 2018

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Diplomová práce

Analýza datových toků v reportovacích nástrojích

Bc. Petr Košvanec

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

6. května 2019

Poděkování

Chci poděkovat Michalu Valentovi za dobré rady a vedení, společnosti Manta a zejména Lukáši Hermannovi za příležitost vypracovat diplomovou práci pod záštitou firmy a usměrňování během celého procesu. Děkuji členům reportingového týmu za cenné informace týkající se ostatních nástrojů. Obrovské poděkování směřuji své rodině, která mě po celou dobu studia podporovala všemi možnými způsoby. Na závěr moc děkuji Danče, která při mně stála za každé situace a je z velké části zodpovědná za to, že jsem se včas přinutil k práci a vše potřebné stihl.

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 6. května 2019 .....

Odkaz na tuto práci

Košvanec, Petr. *Analýza datových toků v reportovacích nástrojích.* Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2019.

Abstrakt

Podstatou této práce je proniknutí do datových struktur v reportovacích nástrojích, které jsou kritickou součástí strategického rozhodování jakékoliv větší firmy. Znalost datových toků pomáhá udržovat kvalitu firemních dat, redukovat dopady změn infrastruktury nebo například dodržovat právní předpisy. Na základě analýzy několika velkých hráčů na poli reportovacího software je v rámci díla vytvořen model reprezentující objekty datových toků. Hlavním cílem implementace bylo vytvořit na základě tohoto modelu prototyp modulu pro software Manta Flow, který se zabývá vizualizací datových toků v širokém spektru technologií. Tento modul, označovaný také jako „konektor“, byl úspěšně vytvořen pro nástroj IBM Cognos a kromě toho, že produkt brzy čeká reálné nasazení, může tato práce sloužit i jako inspirace pro další podobné moduly pro jiné reportovací nástroje.

Klíčová slova business intelligence, data governance, data lineage, datové toky, report, reportovací nástroj, cognos, olap, manta, konektor, obiee, excel, ssrs

Abstract

The essence of this thesis is to understand data structures in reporting tools, which are a critical part of any major firm's strategic decision-making. Knowledge of data flows helps maintain the quality of corporate data, reduce the impact of infrastructure changes, or comply with legislation, for example. Based on the analysis of several large players in the field of reporting software, a model representing data flow objects is created within the work. The main goal of implementation was to create a prototype module based on the model for Manta Flow software, which deals with visualization of data flows in a wide range of technologies. This module, also referred to as a connector, was successfully created for the IBM Cognos tool, and in addition to the fact that the product is about to be deployed and really used, this work may also serve as inspiration for other similar modules for reporting tools.

Keywords business intelligence, data governance, data lineage, data flow, report, reporting tool, cognos, olap, manta, connector, obiee, excel, ssrs

Obsah

Seznam obrázků

Seznam tabulek

Úvod

V dnešní době jsou data zásadní prerekvizitou pro dělání informovaných rozhodnutí při řízení velkých i menších firem a podniků. Aby ale data k tomuto účelu mohla sloužit, musí být zodpovědné osobě podána v přehledné formě a především to musí být data se silnou vypovídající hodnotou, která ideálně už na první pohled dokáží „odvyprávět příběh“.[7] Především k tomuto účelu slouží tzv. reportovací nástroje, které ve formě specifických dokumentů – reportů – vhodně vizualizují data, aby jediným úkolem bylo samotné business rozhodnutí a ne starání se o data v pozadí.

Samotné nástroje ale nemohou zaručit, že zobrazená data budou skutečně kvalitní, pocházející z důvěryhodných zdrojů. Stejně tak není úkolem těchto nástrojů řešit např. právní předpisy, které se na data dané firmy mohou vztahovat, nebo například dopady, které změny v hardwarové infrastruktuře společnosti mohou způsobit.[8] Při řešení všech těchto problémů a mnohých dalších může firmě znatelně pomoci znalost datových toků, tedy schopnost mapovat „průtok“ dat skrze fyzické databázové stroje, čistě logické vrstvy a datové modely tak, aby v každé chvíli byl dohledatelný zdroj proudících dat.

Data během svého životního cyklu typicky proudí skrze několik heterogenních prostředí a jsou zpracovávána množstvím různých technologií. Není možné vyvinout univerzální analyzátor datových toků, který by automaticky zmapoval toky v libovolných technologiích. Naopak je třeba ke každé technologii přistoupit jednotlivě, vyvinout specifický modul a až ten případně začlenit do komplexního řešení. Přesně tento přístup implementuje firma Manta, kde jednotlivé konektory jsou tím, co dokáže porozumět datům ve specifických technologiích. Jeden takový konektor pro business intelligence nástroj IBM Cognos je předmětem této práce.

Kapitola 1

Cíl práce

Téma této diplomové práce je poměrně rozvité, proto chci čtenáře nejprve uvést do světa reportovacích nástrojů, objasnit jejich účel a hlavní motivaci pro jejich korporátní užití. Zároveň je také třeba představit komplexní nástroj Manta Flow, který provádí analýzu datových toků ve vybraných technologiích a objasnit, jakým způsobem se do jeho stávající architektury implementuje nový konektor a co to vůbec konektor a datové toky jsou.

V další části práce mě čeká analýza několika vybraných reportovacích nástrojů. Jmenovitě to jsou IBM Cognos, Oracle Business Intelligence Enterprise Edition, SQL Server Reporting Services a Microsoft Excel. Předmětem rešerže bude především datový model a objekty, se kterými se v daných nástrojích pracuje, architektura jednotlivých prostředí a hlavně datové toky a způsob, jakým obecně tyto nástroje data zpracovávají nebo konzumují. Implementace se bude týkat IBM Cognos, proto tento reportovací software dostane nejvíce prostoru.

Na základě analýzy se pak vyjádřím k myšlence sjednoceného modelu vizualizace datových toků, který zároveň navrhnu a zdůvodním svá rozhodnutí v kontextu objektů, vyskytujících se v analyzovaných nástrojích.

Stěžejním bodem celé diplomové práce bude praktická část – implementace modulu nástroje Manta (konektoru) pro analýzu datových toků v reportovacím nástroji IBM Cognos. Modul bude muset získat z nástroje potřebné informace a metadata a transformovat je do navrženého sjednoceného modelu datových toků tak, aby je nástroj Manta mohl účinně vizualizovat. Cílem není naprogramovat kompletní řešení, protože je očekáváno, že ekosystém Cognos bude značně rozvinutý. Výstupem má být prototyp, který zpracuje a vizualizuje datové toky v rámci vybrané podmnožiny funkcí, které software nabízí.

Samozřejmostí je řádná dokumentace a otestování řešení, jak na úrovni zpracování ukázkových reportů, tak na úrovni funkčnosti zdrojového kódu.

Kapitola 2

Přiblížení problematiky

2.1 Reportingové nástroje

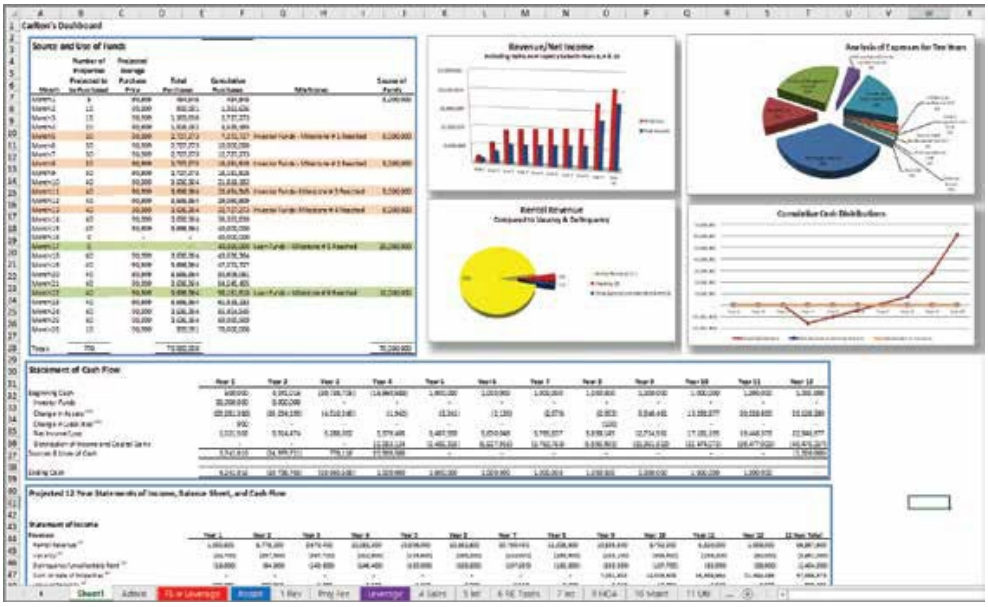
V rámci této sekce popíšu, jaká je motivace pro používání reportingových nástrojů, co všechno nám obvykle software tohoto typu může přinést a jaké podmínky je třeba splnit pro jeho efektivní užívání.

K pochopení účelu reportingových nástrojů je nejprve třeba vysvětlit pojem business intelligence (BI). Tento pojem se sám o sobě stal fenoménem a lze nalézt mnoho i významově odlišných definic. V této práci se přikláním k definici serveru guru99.com: „Business intelligence je soubor procesů, architektur a technologií, které převádějí syrová data na smysluplné informace, které pohánějí obchodní rozhodnutí za účelem zisku. Jedná se o sadu softwaru a služeb, které přeměňují data na použitelnou inteligenci a znalosti. Má přímý vliv na strategická, taktická a operativní obchodní rozhodnutí organizace, přičemž podporuje rozhodování založené na faktech s využitím historických dat spíše než předpokladů a pocitů.“[9]

Obecně by se tak dalo říci, že hlavními přínosy zavedení BI ve firmě jsou:

- informovaná business rozhodnutí
- podpora definování a plnění strategických cílů
- prohloubení znalostí o firemních zákaznících
- podpora produktivity
- pozitivní přispění k udržení přesnosti a kvality dat
- potenciální urychlení návratnosti investic [10]

Reportingové nástroje jsou jedním ze základních prostředků business intelligence. Jejich základním úkolem je vizualizace dat v různých formách. Nejčastěji se jedná o stránkované dokumenty, zvané reporty, které data seskupují do grafů, tabulek, výčtů a mnoha jiných vizualizací, z nichž každá má svůj



Obrázek 2.1: Ukázka reportu v Microsoft Excel [1]

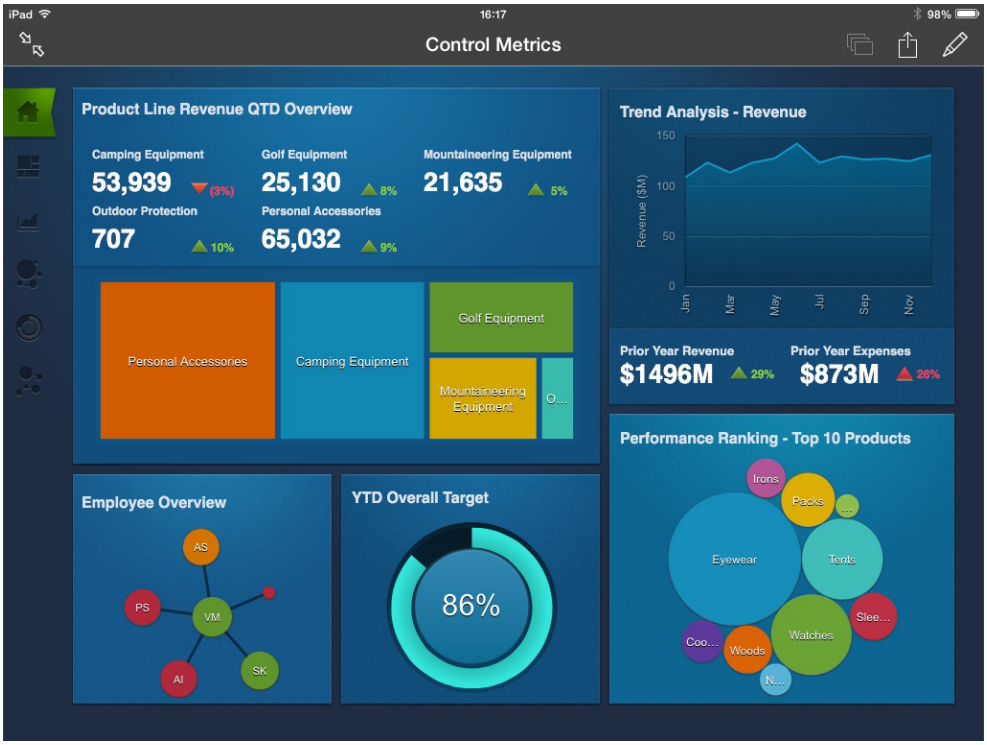
specifický případ užití. Jak bylo již úvodem naznačeno, hlavním cílem těchto nástrojů je data vizualizovat v takové podobě, aby bylo možné na základě prezentovaných informací učinit krátkodobá či dlouhodobá strategická rozhodnutí. [11] Na obrázku 2.1 je ukázán příklad reportu v populárním nástroji Microsoft Excel, který se k reportingovým účelům masivně využívá.

2.1.1 Reporting vs Dashboarding

Důležitým trendem poslední doby je zvyšování míry interaktivity, kterou výsledné reporty poskytují. Někteří zástupci dotahují poskytovanou interaktivitu finálních dokumentů na takovou úroveň, že produkt vlastně ani není určený k tisku. Takové reporty jsou naopak vhodné na volbu vlastních pohledů na data, filtrování dle dat souvisejících a v případě hierarchicky členěných dat i zanořování na detailnější úrovně.

S výše zmíněnými trendy roste i popularita tzv. dashboardů a reportingu „na první pohled“. Řada reportingových nástrojů proto rozlišuje mezi reporty a dashboardy, kde první jmenované jsou více datově orientované. Převažují v nich textově založené, tabulkové či jiné jednoduché vizualizace s důrazem na data samotná. [11]

Je třeba však pamatovat na to, že neexistuje žádný jednoznačný konsensus ohledně toho, co je ještě report a co už by se mělo považovat za dashboard. Jeden takový dashboard vytvořený pomocí IBM Cognos je na obrázku 2.2. Nástroj Cognos ještě vedle reportů a dashboardů nabízí tzv. „interaktivní



Obrázek 2.2: Ukázka dashboardu v IBM Cognos [2]

reporty“, které si berou něco z interaktivity dashboardů, ale zachovávají si robustnost dat a pohledů, které nabízí klasické reporty.

2.2 Analytické nástroje

Řada reportingových nástrojů podporuje přímé propojení s fyzickou databází. Mnohem častěji ale reporty využívají data, která jsou jim poskytována skrze různorodé struktury, nezřídka pojmenované jako datové modely. Výhodou je jednotný přístup k datům díky abstrahování detailů. Běžný uživatel, tedy autor reportů, tak pracuje s připravenými modely, které navíc velmi často použije pro desítky reportů a je s nimi díky tomu už dobře obeznámen. Nevýhodou je, že tyto modely musí typicky vytvořit analytik dobře obeznámený s organizací datového skladu společnosti.

Úzce spojena s analytickými nástroji je technologie OLAP. OnLine Analytical Processing je souhrnné označení pro skupinu technologií, které využívají specifický přístup organizace dat v datovém zdroji k časově efektivnímu dotazování. Základní myšlenkou je strukturovat data způsobem, který bude vhodnější z hlediska očekávaných nebo nejčastějších dotazů. [12] Zároveň je často využíváno dopředných kalkulací, které připraví potřebná data pro poz-

dější využití. Výsledky složitých dotazů mohou být ukládány. Díky tomu je časová odezva dotazů nad daty při vytváření a spouštění reportů založených na datových modelech mnohem nižší a v řádu sekund lze zpracovat a vizualizovat zásadní informace.

Další velkou výhodou je způsob, jakým mohou být data členěna. [13] Ten totiž umožňuje bez další časové zátěži data určitým způsobem filtrovat, nebo se zanořovat do definované detailnější úrovně. Například z pohledu na roční výdej firmy se tak snadno lze dostat na přehled výdejů během jednotlivých měsíců. Stačí jen, aby byl datový sklad vhodně členěn a model nad ním adekvátně vytvořen.

2.2.1 Integrace do reportingových nástrojů

Datové struktury jsou typicky vytvářeny v analytickém nástroji, který může být integrován přímo v uživatelském rozhraní reportingového nástroje. Častěji se ale jedná o samostatný podpůrný nástroj. Například Microsoft SQL Server Analysis Services (SSAS) je analytický software, který je zcela nezávislý na zvoleném reportingovém nástroji, ve kterém budou data v konečném důsledku vizualizována. V tomto případě je trend dokonce opačný – různé reportingové nástroje podporují import dat právě z SSAS. IBM pro svůj nástroj Cognos vytvořila dokonce několik modelovacích nástrojů, kdy každý slouží k specifickému účelu a jen některé jsou přímo integrovány.

Tvorba kvalitního datového modelu obvykle vyžaduje lepší než základní databázové znalosti, slušný přehled o architektuře datového skladu organizace a především poměrně přesnou představu o případech užití dat. Tvůrci reportů proto obvykle nejsou zároveň autory datových modelů, ale mezi těmito rolemi je vyžadována jistá míra součinnosti například právě pro popis běžných případů užití.

2.3 Manta

Manta, celým názvem Manta Tools s.r.o., je původem česká startupová společnost, jejíž hlavním produktem je nástroj pro analýzu a následnou vizualizaci datových toků. Manta původně byla projektem firmy Profinit s.r.o., která od roku 2008 vyvíjela nástroj Manta Flow ve spolupráci s ČVUT v Praze. [14] Projekt svým potenciálem brzy předčil očekávání a zapojil se do grantových programů Technologické agentury České republiky. V roce 2013 se rozhodlo, že vznikne nová firma Manta Tools, která projekt zastřeší. V roce 2014 startup ovládl soutěž „Czech ICT Incubator @ Silicon Valley“, která mu umožnila založit první zámořskou pobočku v San Francisku. V současné době společnost každoročně roste a její produkt využívají ty největší firmy, jmenovitě například PayPal, Vodafone, OBI, Comcast a další. [15]

2.3.1 Datové toky, data lineage

Datový tok v kontextu této diplomové práce je „pohyb dat skrze systémy tvořené softwarem, hardwarem nebo kombinací obojího“, přičemž nejde o množství proudících dat, ale pouze o existenci toku samotného. Schopnost popsat datové toky a znalost původu dat v jakémkoliv bodu jejich životního cyklu se označuje za data lineage. [16] [17]

2.3.2 Graf datových toků

Výsledkem úspěšné analýzy datových toků má být orientovaný graf datových toků. Orientovaný graf je dvojice vrcholů a hran, kde vrcholy jsou neprázdná množina a hrany jsou uspořádané dvojice zmíněných vrcholů. [18]

V kontextu Manta Flow je situace s orientovaným grafem komplikovanější, protože vrcholy i hrany jsou navíc různých typů. Pozitivním zjištěním je fakt, že pro účely implementace konektoru Cognos a se lze efektivně omezit na 3 typy vrcholů a 5 typů hran. Od zbytku typů je konektor Cognos odstíněn množstvím pomocných tříd. Zajímavé typy vrcholů jsou:

- **Node** – reprezentuje konkrétní objekt, který se může potenciálně účastnit datového toku. Trochu matoucí je zaprvé fakt, že tento typ vrcholu se jmenuje node (česky uzel), což je pojem, který se často za vrchol zaměňuje. Za druhé je nutné pochopit že *node* má sám o sobě ještě parametr *nodetype*, který ho odlišuje od uzlů stejné technologie (*resource*). Jedním z hlavních cílů této práce je navrhnout model právě těchto typů vrcholu typu *node* pro reportingové nástroje.
- **Attribute** – reprezentuje dodatečnou informaci, kterou lze přidat[1] k vrcholu typu node. Uživatele může například zajímat umístění objektu v úložišti nebo výraz, ze kterého byl původně objekt vytvořen.
- **Resource** – reprezentuje technologii, kterou Manta Flow zná a dokáže zpracovávat. Například SSRS, Cognos, atp.

Mezi relevantní hrany patří:

- **DirectFlow** – spojuje dva vrcholy typu *node*. Tato orientovaná hrana představuje tok dat ze zdrojového do cílového uzlu.
- **FilterFlow** – spojuje dva vrcholy typu *node*. Představuje nepřímý tok dat ze zdrojového do cílového uzlu. Nepřímý tok vzniká například filtrováním, podmínkami apod.
- **HasAttribute** – spojuje vrchol typu *node* s vrcholem typu *attribute*. Přidaná informace je ve vizualizaci viditelná pro rozkliknutí daného objektu.

[1] „Přidat“ v tomto významu znamená spojit hranou typu HasAttribute

- **HasParent** – spojuje syna s rodičem. Ve vizualizaci je tato vazba vyjádřena zanořením objektu syna do objektu rodiče.
- **HasResource** – spojuje vrchol typu *node* s vrcholem typu resource. Určuje tedy, k jaké technologii datový objekt patří.

Dále je třeba počítat s faktem, že vrchol typu *node* je jednoznačně identifikovatelný v grafu čtveřicí technologie, jméno uzlu, typ uzlu[2] a rodič uzlu. Toho využiji například ve chvíli, kdy budu chtít, aby se uzel s těmito vlastnostmi nevytvářel vícekrát, ale naopak se mapoval na již vytvořený.

Typický postup v implementaci pak bude například vytvoření vrcholů typu *node* s určitými vlastnostmi (atributy). Při vytváření těchto vrcholů bude vyhledán jejich rodič, nebo bude vytvořen v případě, že ještě v grafu neexistuje. Typicky tyto uzly budeme chtít spojit nějakou hranou reprezentující datový tok, tedy hranami přímými (*DirectFlow*) nebo nepřímými (*FilterFlow*). [19]

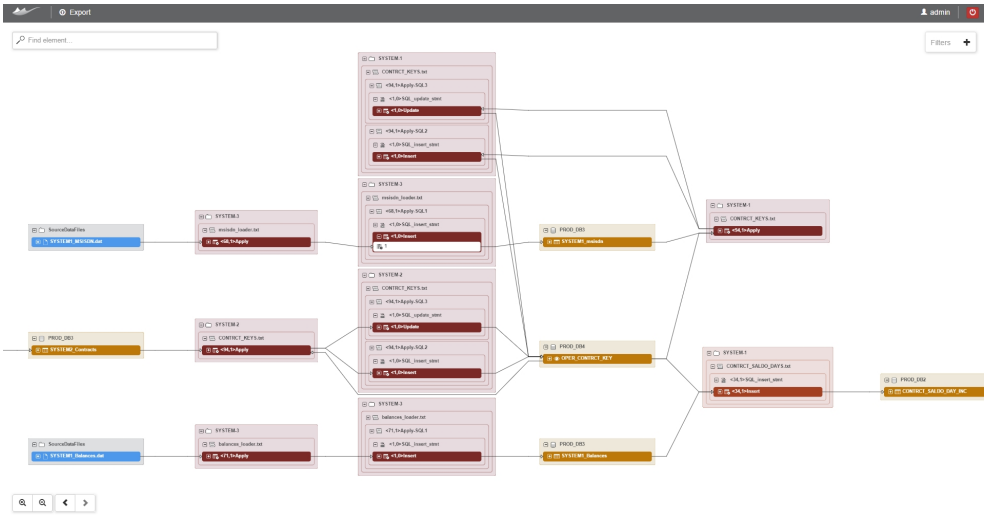
2.3.3 Manta Flow

Jak bylo řečeno, vlajkovou lodí Manty je software Manta Flow. Tento nástroj extrahuje metadata[3], zanalyzuje je a porozumí jim díky permanentně se rozrůstajícímu množství konektorů (skenerů). Typicky jsou tato metadata transformována do modelu specifického pro každou z technologií, která se na toků podílí. Zmíněný model je již v plně v režii Manty. Následně je zpracován do grafu datových toků. Manta Flow tento graf umí zobrazit a uživateli dává dokonce značnou svobodu ve filtrování či konfiguraci výsledné vizualizace. Jedna z možných podob výsledné vizualizace je zobrazena na obrázku 2.3.

Využití znalosti data lineage je široké. Navíc platí, že čím je datová infrastruktura zákazníka komplikovanější, tím jsou možné přínosy znatelnější. Hlavními výhodami použití software Manta Flow mohou být například: [20]

- **Analýza dopadu** – pomáhá identifikovat systémy ovlivněné nebo zasažené zvažovanými změnami. Zpravidla se tedy používá ve fázi plánování změn v infrastruktuře nebo při nepředvídatelné události, která změny ve struktuře dat firmy vyžaduje. [21] Učebnicovým případem užití je například situace, kdy potřebujeme změnit jméno jednoho sloupce v databázi a zajímá nás, do kterých objektů se změna „propíše“, nebo například trochu reálnější situace, kdy hledáme databázové tabulky, které nejsou využívány a chceme je v rámci šetření místa detekovat a odstranit. [22]
- **Analýza příčiny problému** – pomáhá odhalovat místa, která zapříčinila výskyt chyby nebo nekorektních dat. Na rozdíl od analýzy dopadu se inherentně využívá zpětně až po detekování chyby. Data mohou

[2] Nezaměňovat typ uzlu s typem vrcholu!
[3] Metadata jsou doplňková „data o datech“.



Obrázek 2.3: Ukázka vizualizace datových toků v nástroji Manta Flow [3]

procházet různými technologiemi a tedy mnoha vrstvami, ale zajímá nás vždy zdroj (nebo také kořen) chyby. Znalost příčiny chyby umožňuje opravit ji přímo v místě vzniku na rozdíl od běžně praktikovaného přístupu záplatování a redukování symptomů. [23]

- **Přehlednější migrace dat** – při přesunu dat do jiných systémů lze snáze dohledat závislosti mezi daty. To umožňuje se při migraci dat například omezit na určitou jejich podmnožinu, nebo migraci strukturovat do několika nezávislých transakcí. Manta také pomůže odhalit nevyužívaná data a tak snížit potřebnou kapacitu cílového úložného prostoru. Když je datový sklad organizace rozsáhlý, je téměř nemožné manuálně závislosti mezi datovými objekty hledat a je nutné použít automatizované řešení. Manta Flow v tomto případě dokáže zodpovědět důležité otázky jako například „Co ještě zbývá zmigrovat?“, nebo „Jsou zmigrovány všechny kritické části systému?“. [24]

- **Plnění regulatorních požadavků** – právní předpisy, postupy a požadavky na organizaci. V případě neplnění regulatorních předpisů by firmě hrozily pokuty a jiné právní postihy. Velké nároky jsou kladeny na bezpečnost dat a zejména pak na bezpečnost dat uživatelů.[4] Jedním z regulatorních požadavků může být například vymazání všech dat o určité osobě na její požádání. Hledat veškerá data manuálně by opět nebylo za použití rozumných zdrojů možné a tak pomůže znalost datové lineage, kterou Manta Flow nabízí automatizovaně.

[4] V tomto ohledu firmy celosvětově ovlivnila evropská směrnice GDPR.

- **Podpora data governance** – díky velkému množství integrací může zákazník nezištně začlenit data lineage do již existujícího řešení data governance a zvýšit tak důvěru v data a jejich kvalitu. Manta také dokáže mapovat data v některých prostředích, kterým jiné data governance nástroje nerozumí a zlepšit tak celkové pokrytí.

- **Zefektivnění vývoje** – za určitých okolností může znalost datových toků pomoci i zefektivnit vývoj software. Například může jít o agilní vývoj komplexního systému, který spočívá ve využití sandboxů[5] tak, aby nebyla ohrožena ostatní funkčnost systému. [25] Vytváření těchto sandboxů vyžaduje obvykle manuální zdlouhavou práci při hledání závislých objektů a procedur. Zavedení Manta Flow do tohoto procesu pomůže proces hledání závislostí mnohonásobně urychlit.

Z výše uvedeného výčtu vyplývá, že nejsilnějším artiklem Manty je právě automatizace činností, které dodnes lidé ve špičkových organizacích dělají manuálně. Z business pohledu je proto Manta Flow velmi lákavým softwarem především pro středně velké a velké firmy, jejichž datová infrastruktura je příliš komplexní na manuální analýzu. Takovým subjektům dokáže tento produkt ušetřit velké množství finančních i lidských zdrojů.

Mezi technologie, které Manta Flow dokáže v současné době analyzovat, patří například databázové systémy Oracle, IBM DB2, PostgreSQL, programovací jazyky Java, PigLatin, SQL a mnohé další datově orientované nástroje. Manta se přizpůsobuje poptávce a proto aktivně probíhá i vývoj konektorů pro reportovací nástroje jako jsou IBM Cognos, Microsoft Excel, Microsoft SSRS a jiné.

2.3.3.1 Integrace datových toků do nástrojů třetích stran

Zákazníci pochopitelně chtějí homogenní vizuální prostředí a používání co nejmenšího množství podpůrných nástrojů. Proto je důležitým faktorem také schopnost integrace získaného grafu datových toků do data governance nástrojů třetích stran, jako jsou například Informatica, Collibra DGC, IBM IGC. Manta Flow Server tato data poskytuje v rámci svého API[6] ve formátu JSON[7].

Požadavky na integraci jsou jednou ze zásadní motivací pro vytvoření společného modelu pro datové toky v reportovacích nástrojích, což je jedním z cílů této práce. Čím více budou reportovací nástroje využívat společných typů uzlů, tím méně mapování, lidských a finančních zdrojů si vyžádá transformace grafu do libovolného nástroje třetí strany.

[5] Sandboxy jsou bezpečnostní prvky, které brání vyvíjenému a netestovanému kódu způsobit trvalé nežádoucí účinky na okolní systémy.

[6] API (application programming interface) je sada funkcí, které jsou určeny k využití zejména externími programy.

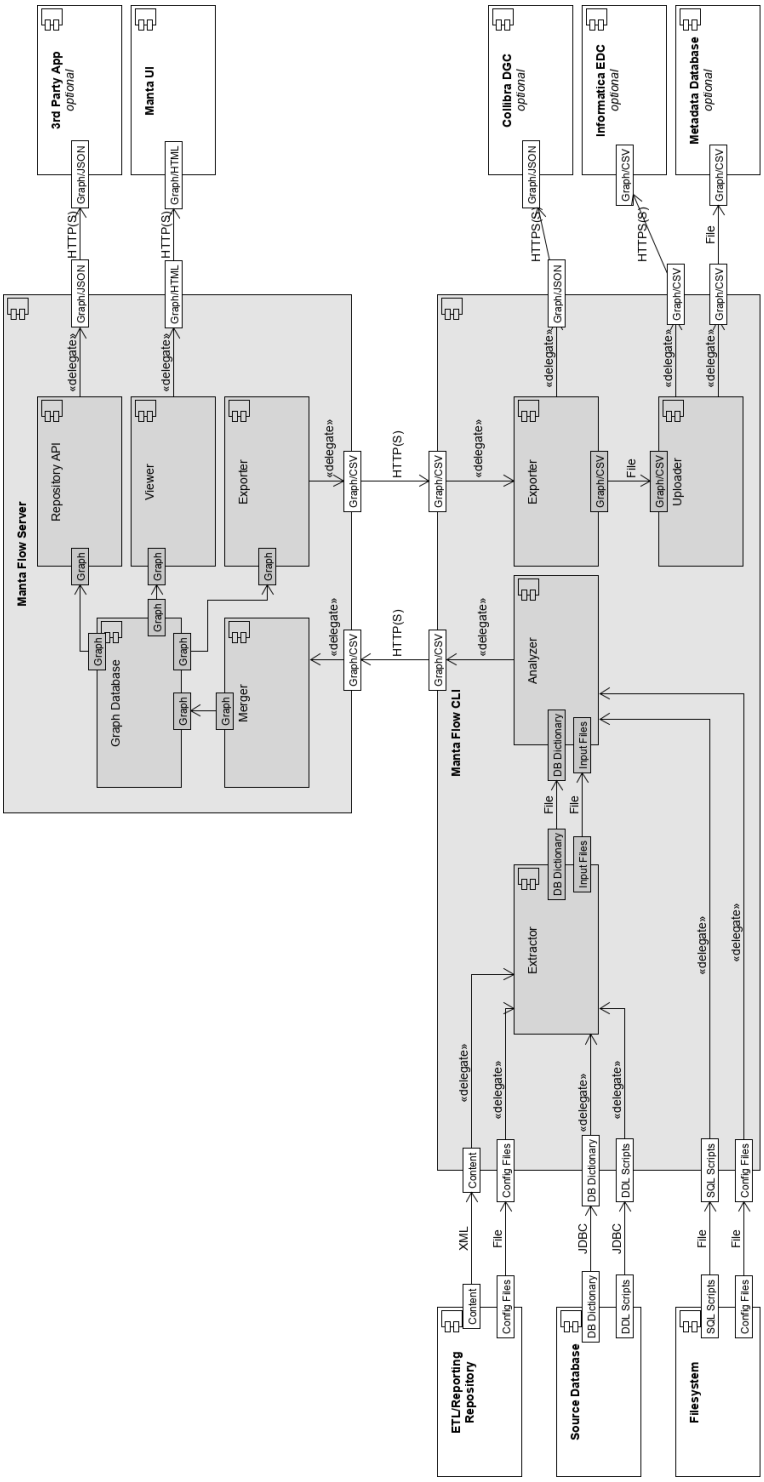
[7] JSON (JavaScript object notation) je formát dat určený pro jejich přenos.

2.3.3.2 Architektura Manta Flow

Obrázek 2.4 zachycuje architekturu nástroje, jeho komponenty a interakci s prvky třetích stran. Software Manta Flow se skládá ze dvou hlavních a jedné podpůrné komponenty. [4]

- **Manta Flow CLI** – jedná se o Java aplikaci běžící v příkazové řádce, která extrahuje všechna potřebná vstupní data ze zdrojových databází, skriptů a specifických úložišť, zanalyzuje je a exportuje získaná metadata ve formě grafu do dedikovaného Manta Flow Serveru. Metadata může také volitelně poskytnout aplikacím třetích stran.
- **Manta Flow Server** – serverová aplikace také psaná v Javě. Účelem serveru je ukládat („mergovat") v optimální formě všechna získaná metadata uvnitř tzv. metadata repository ve formě grafové databáze. Tato data pak exporter transformuje pro import do cílové aplikace třetí strany, nebo připraví pro zobrazení ve webovém uživatelském rozhraní.
- **Manta Flow Service Utility** – podpůrná aplikace, která pomáhá při konfiguraci a aktualizuje uživatelské rozhraní. Využívá klientem poskytnuté konfigurační soubory umístěné v CLI.

Z této architektury a zadaného úkolu vyplývá, že požadovaný konektor pro nástroj IBM Cognos bude sada modulů pro Manta CLI. Konkrétně bude třeba řešit extrakci dat, transformaci na interní model a vygenerování grafu datových toků z tohoto modelu.



Obrázek 2.4: Architektura Manta Flow [4]

Kapitola 3

Analýza

V této kapitole jsou rozebrány čtyři různé reportingové nástroje. Pro všechny z nich je momentálně vyvíjen nebo již vyvinut konektor pro nástroj Manta Flow. Analýza se zaměří na datové objekty, které jsou definovány uvnitř specifikací reportů a různých analytických a logických modelů. Úkolem je rozebrat význam těchto objektů pro datové toky v daném nástroji, aby bylo možno identifikovat napříč nástroji významově podobné objekty. Není klíčové identifikovat všechny vazby mezi nimi.

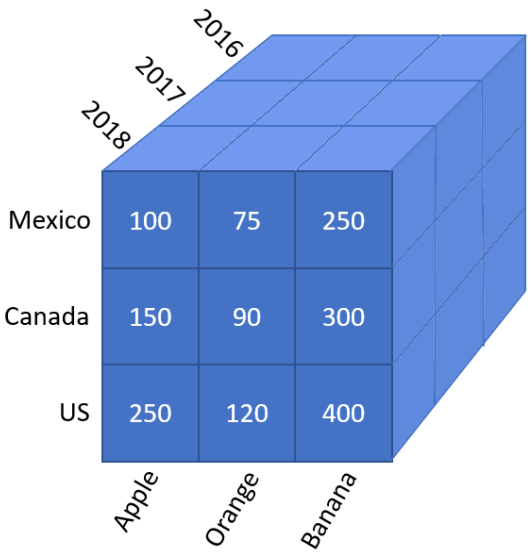
Společným rysem analýzy ve všech těchto nástrojích je potřeba přístupu ke zmíněným specifikacím. Reportingové nástroje mívají typicky vlastní úložiště reportů, ve kterých jsou reporty a další zásadní objekty uložené ve formě strukturovaných souborů, nejčastěji formátu XML[8] nebo JSON.

Analýza SQL Server Reporting Services na rozdíl od ostatních obsahuje i detailnější pohled do struktury jednotlivých typů vizualizací pro ilustraci těchto objektů. Ostatní nástroje už místo výčtu těchto prvků používají abstraktnější označení *Report item* zahrnující všechny tyto prvky reportů. Analýza pro IBM Cognos je záměrně mírně obsáhlejší, protože tento nástroj bude předmětem implementace. Zbytek nástrojů je potřeba analyzovat jen do té míry, aby mohla být vytvořena společná reprezentace, tedy sada uzlů vycházející ze sémanticky podobných objektů.

3.1 OLAP kostka

Společným rysem analyzovaných nástrojů je podpora OLAP kostek jako datových zdrojů. Ty jsou základním stavebním kamenem pro OLAP analýzu. Jedná se o N-dimenzionální krychle, častěji označované jako kostky. Dimenze představuji dodatečnou informaci blíže popisující tzv. fakt. Fakt bývá nejčastěji číselná hodnota, která může vyjadřovat příjmy nebo výdaje, počet pro-

[8]Extensible markup language je značkový metajazyk, který slouží k vytváření jiných značkových jazyků.



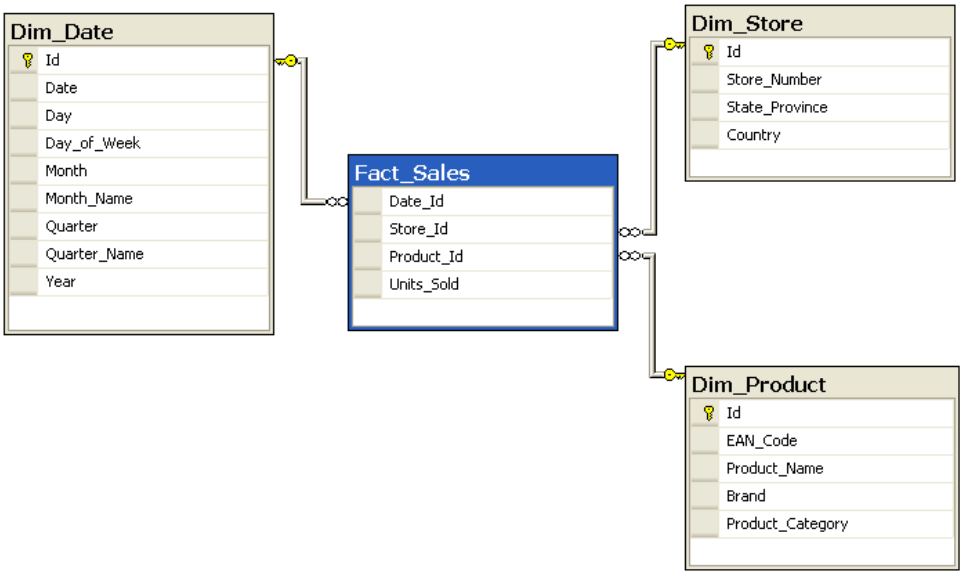
Obrázek 3.1: Příklad kostky [5]

daných kusů a podobně. Dimenze pak mohou představovat například různé časové úseky, prodejní místa, druh prodávaného produktu. Jeden takový příklad ilustruje obrázek 3.1.

Kostky jsou vhodné pro provádění několika operací.[26] Uvedu anglická pojmenování těch zásadních z nich, protože české verze se používají minimálně.

- **Slicing (krájení)** – omezení se na podmnožinu členů dimenze o jednom prvku. Například volba určitého roku prodeje nebo výrobku.
- **Dicing (kostkování)** – omezení se na podmnožinu členů dimenze o více prvcích. Například volba období od určitého roku nebo skupiny výrobků.
- **Roll up** – změna detailu o úroveň výš. Např. místo prodaných kusů za měsíc chceme prodané kusy během celého roku.
- **Drill down** – změna detailu o úroveň hlouběji. Např. místo prodaných kusů v rámci jednotlivých roků chceme prodané kusy během jednotlivých měsíců.

Prerekvizitou pro vystavění kostky je jedno ze specifických schémat tabulek v databázi. V praxi se běžně používá buď star schéma (hvězdicové), nebo snowflake schéma (vločkové). Oba názvy vychází z vizuální podoby diagramu popisujícího tabulky v databázi. V pomyslném středu schématu je faktová tabulka obsahující daný fakt (measure) a cizí klíče do všech dimenzí specifikujících fakt. Snowflake schéma navíc dovoluje členit dimenze tak, aby byly



Obrázek 3.2: Ilustrace star schematu [6]

tabulky normalizované.[27] To sice sníží redundanci dat a ušetří diskový prostor, ale dotazování je složitější a údržba databáze také. Příklad star schematu popisujícího počet prodaných produktů v určitém časovém rozmezí v různých prodejnách je ukázán na obrázku 3.2.

Kostky velmi často bývají zdroji dat v reportingových nástrojích kvůli svému primárnímu účelu rychle odpovídat na komplexní dotazy. Výsledky těchto dotazů jsou pak v reportech prezentovány uživateli vhodnými vizualizacemi. Všechny z analyzovaných nástrojů kostky podporují.

3.1.1 Objekty v kostce

Přestože existuje mnoho různých technologií implementujících kostky, jejich struktura je vždy velmi podobná, což z objektů uvnitř kostky automaticky dělá kandidáty na uzly společného modelu. Napříč analytickými nástroji pro tvorbu kostek se opakují následující objekty:

Dimension reprezentuje jednu dimenzionální tabulku.

Attribute koresponduje s jedním ze sloupců tabulky dimenze.

Hierarchy člení atributy dimenze do jednotlivých levelů. Může existovat více hierarchií, tedy způsobů členění, pro jednu dimenzi.

Level určuje kolekci atributů dimenze. Používá se pro roll up/drill down operace.

Measure jsou odvozeny od sloupců faktové tabulky. Tyto číselné údaje bývají vyhodnoceny v kontextu dimenzí.

Calculated member jsou předem připravené výpočty na základě ostatních členů kostky, které mohou být přepočteny v době dotazu.

3.2 SQL Server Reporting Services

Hlavním zdrojem informací pro vypracování této kapitoly byla interní dokumentace firmy Manta vycházející ze znalostí získaných při implementaci konektoru pro SSRS Jaroslavem Kotrčem.[28]

SQL Server Reporting Services (SSRS) je systém firmy Microsoft určený na generování a správu reportů a souvisejících objektů během jejich životního cyklu. Samotná tvorba reportů však probíhá ve specializovaném nástroji Report Builder. Aplikace SSRS, kterou uživatel ovládá skrze webové uživatelské rozhraní, nemá téměř žádné editovací schopnosti reportů a například tak umožňuje měnit pouze jméno reportu, ale ne jeho podobu. Uživatelům pomáhá spravovat různé druhy interaktivních reportů, publikovat, distribuovat a připravovat jejich datové zdroje. SSRS je jedna z částí sady Microsoft SQL Server services, mezi které dále patří Analysis services (SSAS) a Integration services (SSIS).

SSIS je platforma pro stavbu a správu dat tak, aby data byla kvalitní, požadovaného formátu a optimalizovaná pro vysoký výkon. Jedná se o typického představitele extract-load-transform software. Produktem SSIS jsou balíčky (*Package*), které pohybují s daty a mění jejich podobu dle požadavků. Nástroj pracuje na fyzické vrstvě a jedním z jeho úkolů je připravit datové zdroje pro analýzu a reporting. [29]

SSAS je posledním ze zmíněné trojice software. Jeho účelem je vytvořit analytické pohledy na data v datových skladech, často upravená pomocí SSIS. [30] Tyto analytické pohledy nástroj reprezentuje jako tabulární modely nebo OLAP kostky. Tyto kostky mají velmi podobnou strukturu, jaká byla nastíněna v sekci 3.1.1.

Definice reportů a datových zdrojů šlo z aplikace SSRS získat dříve z interní MSSQL databáze klasickým SQL dotazem nad tabulkou *dbo.catalog*. Tento způsob narážel na omezení u extrakce definic datových zdrojů. Jakékoliv změny v jejich definicích se ukládaly jiným způsobem než originál a nebylo možné metadata v nich obsažená získat. Informace pro připojení k datovým zdrojům se tak musely v určitých případech zadávat manuálně. Od verze SSRS 2017 je na extrakci všech potřebných informací možné využít REST[9] rozhraní, které těmito problémy netrpí. Extrakci však komplikuje fakt, že se reporty mohou vnořovat a v takových případech je pro analýzu kořenového reportu získat

[9]REpresentational State Transfer je protokol popisující sadu pravidel pro tvorbu bezestavových webových služeb.

i definice reportů do něj vnořených. Oba zmíněné způsoby extrakce umožňují získání konkrétní definice reportu za použití ID reportu.

SSRS reporty používají pro výběr dat z SSAS kostek jazyk MDX[10]. Tento jazyk podobný SQL také obsahuje řadu funkcí pro kalkulace a filtrace nad těmito daty. Pro zpracování tohoto jazyka a zachycení potřebné sémantiky je v konektoru SSAS použit parser[11] používající gramatiku tohoto jazyka, která musela být dle specifikací jazyka vytvořena. Samotný parser je postavený na knihovně ANTLR. Obdobně mohou obsahovat i DAX výrazy pro výběr dat z tabulárních modelů, případně čisté SQL pro výběr přímo z databází.[31]

3.2.1 Datově relevantní objekty SSRS

V následujícím výčtu jsou uvedeny objekty nalezené ve specifikacích reportů a objekty související. Zde u nástroje SSRS ještě uvádím i příklady jednotlivých vizualizací, které jsou v reportech k nalezení. Dále v práci budou většinou jednotlivé vizualizace spadat do kategorie *Report item*.

Server představuje jednu instanci SSRS. Je to kořenový objekt celé hierarchie. Reporty jsou v něm uloženy v rámci adresářové struktury. Ve vizualizaci grafových toků budou tyto složky pravděpodobně vynechány.

Report je společným označením pro všechny zmíněné typy reportů, které SSRS spravuje. Zajímavé je, že report může obsahovat jiné reporty, které jsou obaleny v kontejneru s označením *subreport*. Pro účely generování datových toků tak subreporty nejsou nijak kritické. Každý z reportů, který se podílí na výsledné podobě dokumentu, má vlastní specifikaci.

Data set reprezentuje data navrácená jako výsledek dotazu vůči externímu datovému zdroji. Součástí jeho definice je element *Query* s daty specifickými pro daný dotaz, výsledné sloupce označené jako *Field*, parametry a případně použité filtry.

Field je prvek *Data setu*. Koresponduje se sloupcem výsledku použitého dotazu.

Chart je prvním příkladem vizualizace v SSRS. Jedná se o grafy, které se odkazují na data ve formě datových řad (*Data series*). Každá datová řada obsahuje ještě jednotlivé hodnoty označené jako *Data point*.

Gauge panel je označení pro sadu vizualizací, které fungují na principu znázornění určitého postupu nebo stavu vůči kompletnímu celku. Jako

[10] MDX znamená MultiDimensional eXpressions, neboli multidimenzionální výrazy.

[11] Parsery jsou nástroje pro rozpad složitých řetězců na jednotlivé elementy odlišného významu pro další zpracování.

gauge panel je tak definován například ProgressBar[12], nebo vizualizace připomínající tachometr auta. Gauge panel má pod sebou v XML specifikaci ještě řadu elementů, např. *Gauge pointer*, *State indicator*, *Gauge scale*. Podstatné ale je, že na nejnižší úrovni v listech pod těmito objekty se vyskytují *Tooltip* a *Value* obsahující zobrazovaná data.

Map je další z druhů vizualizace připomínající geografické mapy. Podobně jako *Gauge panel* má pod sebou rozvětvenou strukturu potomků a na nejnižší hladině se znovu vyskytují *Tooltip* a *Value*.

Textbox reprezentuje jednoduchou oblast s textem, který může být staticky vyplněný (a tedy pro datové toky nezajímavý), ale i pocházet z databáze. Přímo pod tímto elementem lze nalézt už zmíněné *Tooltip* a *Value*. Typicky se vyskytuje v kontejneru *Tablix*.

Value, Tooltip, Label, Data point jsou prvky, které přímo obsahují odkazy na data umístěna v *Data setech*. Přestože v SSRS je jejich význam vzájemně odlišen, pro účely sjednoceného modelu by mohly všechny být reprezentované jedním typem uzlu, protože se v zásadě vždy jedná o nějakou hodnotu.

Parameter je proměnná použitá pro parametrizaci reportu. Report díky parametrům může například měnit jazyk zobrazení, podmiňovat formátování prvků reportu na základě hodnoty (*Value*) proměnné a filtrovat zobrazená data. Parametr dále obsahuje ještě *Label*.

Objects reprezentují vestavěné objekty ve specifikacích reportů, které obsahují hodnoty jako autorovo jméno, jméno reportu, čas vzniku reportu atp. V hierarchii XML elementů jsou seskupeny do kolekcí, které *Report* obsahuje. Tyto objekty bývají staticky vyplněné, proto do nich nevedou datové toky z databází a jiných zdrojů, ale často jsou na datech v nich obsažených závislé vizualizace, proto je dobré je také analyzovat.

3.3 Oracle Business Intelligence Enterprise Edition

Analýza Oracle Business Intelligence Enterprise Edition (OBIEE) je inspirována interní dokumentací, kterou pro konektor ve firmě Manta OBIEE sepisuje Yauheniy Buldyk.[32]

Oracle Business Intelligence Enterprise Edition (OBIEE) je business intelligence nástroj od společností Oracle. Zákazníkům pomáhá v rychlejších a informovanějších obchodních rozhodnutích poskytováním vizuální analýzy dat. Jedná se o jednu ze standardních BI platforem, která přináší celou škálu

[12] ProgressBar je obvykle obdélníkový ukazatel aktuálního stavu procesu. Typicky ukazuje procenta pokroku.

možností pro interaktivní dashboardy, mobilní analytiku, řízení firemních strategií atd. Tvorba reportů ve většině ostatních BI nástrojů je časově náročná. Pokud je třeba platforma, která dokáže odstranit nadbytečné kroky při generování vytváření analýz reportů, zákazníci často volí OBIEE. [33]

Reporty, datové modely a další objekty jsou uložené v rámci tzv. *Presentation catalogu*. Pro extrakci těchto objektů z katalogu jsou k dispozici webové služby založené na protokolu SOAP[13]. Tyto služby využívají session[14]. To znamená, že se nejdřív musí zavolat služba na autentikaci a v případě validního přihlášení je až pak možné volat službu pro získání objektů z katalogu.

V jiných reportovacích nástrojích můžeme nalézt různé typy reportů. V OBIEE jsou reporty sjednocené, ale existují různé druhy layoutů, které mají v případě tohoto nástroje větší význam než u ostatních. Layouty totiž mohou být definovány sedmi různými způsoby, mezi které patří například RTF[15], XLS 3.4 a zejména XPT, což je souborový formát, ve kterém ukládá definice reportů nástroj BI Publisher Layout Editor. Je založený na jazyku XML. Zároveň je tento druh layoutů jediný, který je momentálně ve firmě Manta zpracováván, proto analýzu datově relevantních objektů omezím právě na něj.

3.3.1 Datově relevantní objekty OBIEE

Následuje výčet objektů relevantních pro datové toky v OBIEE. Jejich hierarchickou strukturu demonstruje obrázek 3.3.

Server podobně jako u SSRS představuje běžící instanci OBIEE. V kontextu datově relevantních objektů představuje kořen celé hierarchické struktury a úložiště, které obsahuje všechny ostatní objekty.

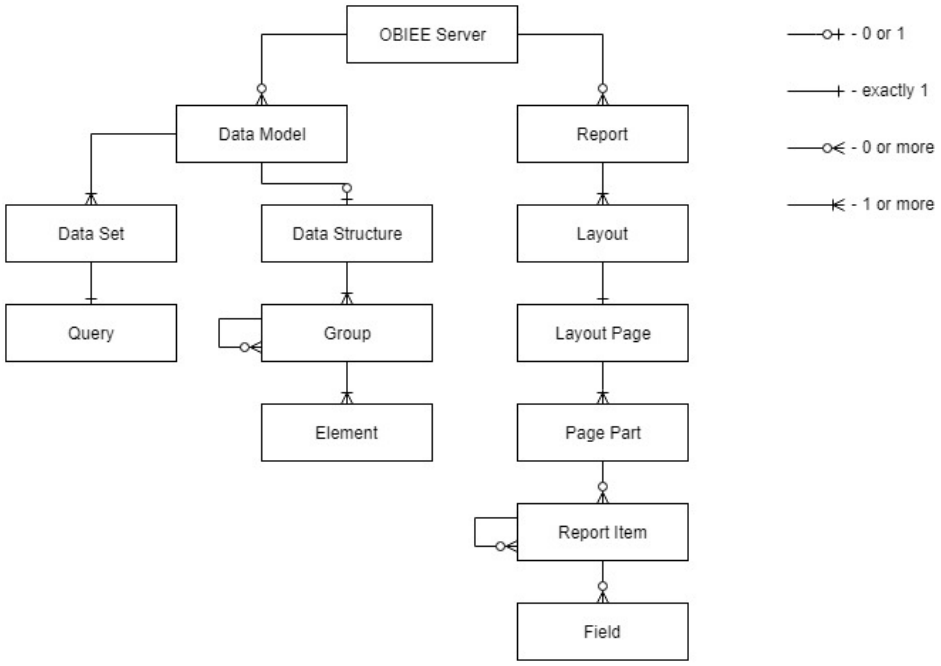
Report je dle očekávání hlavním produktem. Zajímavým faktem je, že specifikace reportu neobsahuje žádné vizuální prvky. Obsahuje pouze *Properties*, tj různé proměnné reportu, a dále jen odkazy na *Data model*, *Layouty* a případné šablony stylů a reportu. Tyto závislosti jsou ale uloženy i extrahovány samostatně. V jistém smyslu je tak extrakce podobná jako u SSRS a jeho vnořených reportů.

Layout představuje kořenový kontejner pro všechny viditelné prvky reportu. Je definován ve vlastní specifikaci. Je podporováno mnoho typů uložení specifikace, zde jsou potomkovské objekty určené specifikací typu xpt. Report může tvořit více těchto Layoutů a ani jejich typy nemusí být shodné.

[13] Simple object access protocol je protokol pro zasílání zpráv XML formátu.

[14] Session je dočasné síťové spojení pro výměnu zpráv.

[15] Rich Text Format je platformně nezávislý formát souborů od Microsoft, který umožňuje ukládání textových souborů s řadou formátovacích příkazů.



Obrázek 3.3: Hierarchická struktura objektů OBIEE [32]

Layout page je stránka uvnitř layoutu. V rámci analyzovaného layouzu xpt formátu je možná jen jediná stránka v layoutu. Vícestránkové reporty jsou tedy podporovány jedině ve formě využití více layoutů.

Page part, Layout grid, Repeat section jsou všechno kontejnery s heterogenní vnitřní strukturou, které v sobě obsahují kolekce různých report itemů. Samotné datové toky neovlivňují, ale poskytují jmenný prostor pro obsažené objekty.

Report item je jako u ostatních reportovacích nástrojů abstraktní označení pro různé druhy vizualizací. I v OBIEE mají různou vnitřní strukturu a platí, že v listech této stromové struktury se dají detekovat datové toky. Níže je uveden jen výčet druhů report itemů použitý v OBIEE, detailní strukturu není třeba rozebírat.

- *DataTable*
- *DataList*
- *PivotTable*
- *Chart*
- *Gauge*
- *Image*

- *Textbox*

Field[16] se nachází v listech hierarchické struktury *Layoutu* uvnitř jednotlivých *Report itemů*. Představuje referenci na *Elementy*.

Data model je struktura, která obsahuje sadu instrukcí pro nástroj BI Publisher k získání dat pro reporty. Výsledky těchto instrukcí jsou popsány v rámci přidružené *Data structure*. Modely jsou samostatné objekty v rámci serveru (v katalogu). Tento objekt může být značně komplexní, sdružovat data z několika datových zdrojů a různými způsoby je ještě agregovat a vytvářet nad nimi dodatečné výpočty.

Data set je sada dat typicky získaná jako výsledek nějakého dotazu (*Query*) nad datovým zdrojem. Svou formou tyto data připomínají databázové tabulky, od kterých se také často skutečně odvozují. Data sety mohou sbírat data z různých zdrojů, jak je popsáno v předchozí sekci.

Query obsahuje definici dotazu, který je použit pro extrakci dat z datového zdroje. Každý datový set má přiřazen právě jeden *Query objekt*.

Group se vztahuje vždy k jedinému *Data setu*, přičemž už konkrétně zobrazuje podobu získaných dat v rámci vnořených *Elementů*. Při změně *Data setu* nebo závislého *Query* se v tomto objektu data automaticky aktualizují. Grupy se mohou také vzájemně vnořovat, což ovlivní jména elementů vnořené grupy a tím i výraz pro odkazování se na tyto elementy.

Element je potomkem grupy a reprezentuje jeden konkrétní sloupec výsledku dotazu nad datovým zdrojem. Odkazují se na ně *Fieldy* uvnitř jednotlivých prvků reportu (*Report itemů*).

3.4 Microsoft Excel

Při vypracování této kapitoly jsem se inspiroval souběžně vypracovávanou diplomovou prací kolegy Daniela Míčka. [34]

Microsoft Excel je součástí kancelářské sady Microsoft Office. Jedná se primárně o tabulkový editor pro zaznamenávání a analýzu převážně numerických dat, který je široce využíván i k reportingovým účelům.[35] Výsledný dokument obsahuje několik listů stejné struktury. Data v nich jsou členěna do sloupců a řádek, které tvoří buňky.

Do jednotlivých buněk lze kromě klasických hodnot umístit matematické vzorce, které obvykle využívají vestavěné nebo uživatelem definované funkce. Ve vzorcích jsou zpravidla umístěny reference na jiné buňky, které v tomto

[16] Pozor na záměnu s Fieldem u SSRS. Ten je definován na logické vrstvě, zde je Field užit vy významu buňky nebo pole.

kontextu slouží jako zdrojová data pro výpočet. Každou buňku nebo rozsah buněk lze jednoznačně identifikovat adresami sloupců a řádků. Je možné ve vzorci odkazovat se na data umístěné v jiném listu či sešitu.

Pro uložení excelovských dokumentů používá sada Microsoft Office formát Office Open XML (OOXML). Soubor uložený na disku má nejčastěji příponu xlsx[17] a jedná se o adresář komprimovaný ve formátu zip[18]. Obsahuje soubory společné pro všechny nástroje sady a samozřejmě také soubory typické pro Excel umístěné v podadresáři *xl*. Tyto soubory jsou uloženy ve formátu XML[19] a obsahují specifikace různých objektů, které se podílí na výsledné podobě uspořádání dat dokumentu.

Microsoft Excel neposkytuje žádné centralizované úložiště dokumentů. Nástroj nechává úložiště plně v rukou uživatele. V ideálním případě má proto firma vlastní dedikované úložiště s definovanou strukturou, kam autoři excelovských sešitů mohou soubory ukládat. Kvůli tomu konektor pro Excel nemusí řešit extrakci tak, jako to řeší ostatní konektory. Vstupem pro konektor budou složky, ve kterých má Excel dokumenty najít. Program tak musí řešit jen rozbalení xlsx souboru a načtení získaných XML specifikací.

Tento reportingový nástroj pochopitelně podporuje externí zdroje dat, tj. dedikované databáze i soubory vybraných sloupcových formátů. Informace o těchto datových zdrojích jsou umístěny v souboru *xl/connections.xml*. Samotné dotazy pro získání dat z definovaných zdrojů Excel jsou uloženy v jedné ze dvou variant.

1. **Microsoft Query** – vestavený nástroj pro definici dotazů. Používá dotazovací jazyk dle daného externího zdroje. Ukládá celé dotazy a umožňuje automatické aktualizace při změně dat ve zdroji. V základu podporuje řadu ovladačů a umožňuje i spojení skrz ODBC[20] či dodané ovladače.

2. **Power Query M** – v novějších verzích nahrazuje Microsoft Query. Tento samostatný nástroj, který je v rozšířených verzích balíku Office dodáván, používá vlastní dotazovací jazyk nazvaný *M*. Nástroj umí kombinovat dotazy nad různými zdroji dat a vytvořit tzv. Data Mashup.

3.4.1 Datově relevantní objekty Excelu

V této sekci budou rozebrány jednotlivé datově zajímavé objekty, které se účastní datových toků v nástroji Excel. Specifikace řady z nich jsou uloženy jako samostatné soubory v adresáři *xl*.

[17] Používají se ještě další verze xls souborů, ale pro účely této postačí analýza xlsx.

[18] Zip je formát pro archivaci a kompresi dat.

[19] Typ uložených objektů je dále upřesněn souborem *[ContentTypes].xml*, který je také umístěn v xlsx souboru.

[20] Open Database Connectivity je rozhraní vyvinuté Microsoftem pro přístup k datům v databázích za použití SQL.[36]

Workbook (sešit) představuje excelovský dokument. Obsahuje odkazy na všechny v něm umístěné listy. Může také obsahovat informace o objektech, které jsou v rámci listů sdílené, např. cache kontingenční tabulky, referencované externí soubory, definované názvy.

Worksheet (list) obsahuje informace o datech v neprázdných buňkách. Tyto informace jsou uloženy po jednotlivých řádcích.

Cell (buňka) může obsahovat různé typy, např. číselnou hodnotu, řetězec, boolean[21]. Ukládá také informaci o případném vzorci a poslední vypočtené hodnotě.

Graph používá data ve formě datových řad, tabulek nebo kontingenčních tabulek a vizualizuje je. Excel nabízí značné množství různých typů grafu. Výhodou je, že všechny dodržují velmi podobnou definovanou strukturu XML. Grafy se mohou vyskytovat na vlastním typů listů.

Data series (datová řada) je kolekce souvisejících hodnot, které jsou často umístěné v sousedních buňkách.[37] Jsou jedním z nejběžnějších zdrojů dat pro grafy.

Native table patří mezi definované objekty. Využívá data umístěna ve sloupcích. Vytvořit ji lze ručně označením oblasti dat, nebo automaticky při importu dat z externího zdroje. Názvy nativních tabulek musí být unikátní v rámci celého sešitu, aby na ně šlo odkazovat strukturovanou referencí.

Query table (dotazová tabulka) je dvourozměrná tabulka reprezentující výsledek dotazu z externího zdroje. Tento druh tabulky nelze v nástroji svobodně využívat, vzniká a je udržován automaticky.

Static table představuje obdélníkovou oblast dat, která byla ručně zadána uživatelem. Ve specifikacích se tyto tabulky nevyskytují, je třeba je v listech identifikovat a zavést uměle.

Pivot table[22] je základním nástrojem k porovnání dvou typů dat.[38] Data jednoho typu jsou zobrazena ve sloupcích a data druhého v řádcích. Pivotové tabulky nelze vlastnoručně vyplňovat, protože jsou založené na datech již importovaných nebo vlastnoručně definovaných. Data pro tabulky se ukládají do *pivot cache*.

Pivot cache reprezentuje objekt, který udržuje repliku části datového zdroje. Není viditelná a je vždy spojená s pivotovou tabulkou a daným zdrojem. [39] V případě změn zdrojových dat je potřeba explicitně aktualizovat kontingenční tabulku, čímž dojde k aktualizaci její cache a tím i sumárních hodnot výsledné kontingenční tabulky.

[21] Booleanovské hodnoty jsou true a false, neboli pravda a nepravda.
[22] Někdy také označovaná jako kontingenční tabulka.

Defined name (definovaný název) představuje alias pro vzorec nebo jeho část. Používá se pro zpřehlednění komplikovaných vzorců a explicitní pojmenování tabulek.

External connection (externí datové spojení) obsahuje informace o použitém databázovém zdroji včetně connection stringu[23] a commandů, které obsahují SQL dotazy.

External link (data externího sešitu) představuje informace o použitých dat pocházejících z jiného sešitu. Informace o nich se uchovávají v souborech umístěných v adresáři *xl/externalLinks*. Data z použitých listů jsou v nich fyzicky zkopírována.

Slicer (Průřez) a slicer cache filtrují obsah tabulek. Slicer obsahuje definici filtru a cache uchovává potřebná metadata pro filtrování.

Pro lepší pochopení závislostí mezi některými z těchto objektů uvádím diagram rozhraní (3.4) pro konektor Excel vytvořený v Mantě Danielem Míčkem.

3.5 IBM Cognos

Cognos je webově orientovaná sada business intelligence softwarových nástrojů, vyvíjená firmou IBM, která původní společnost koupila v roce 2008. Tento „čistokrevný“ reportingový nástroj byl navržen s ohledem na netechnické pracovníky, aby jim umožnil získávat data, analyzovat je, monitorovat různé metriky a produkovat reporty pro předpověď trendů trhu a podporu business rozhodnutí.[40]

Specifikace reportů a modelů, balíčky importovaných dat, informace o užitých datových zdrojích a další zásadní soubory jsou ukládány v úložišti zvaném *Content store*. Jedná se o úložiště založené na relační databázi. K přístupu do Content store se využívá *Content manager*, což je služba využívající JDBC[24] API. [41] Příštup k této službě je umožněn v rámci Cognos SDK[25], které má i svůj Java balík. Využití SDK bude zásadní pro extrakci specifikací, protože alternativní způsoby, mezi které patří například REST rozhraní běžícího serveru Analytics[26], jsou zastaralé a nedostačující.

Pro identifikaci objektů uvnitř Content store se používá *search path*. Jedná se o jazyk připomínající XPath[27], který využívá vztahů rodič-potomek napříč adresáři. Jakékoliv odkazy na externí specifikace jsou uvedeny právě formou search path. Důležitým úkolem pro konektor bude umět tuto search path

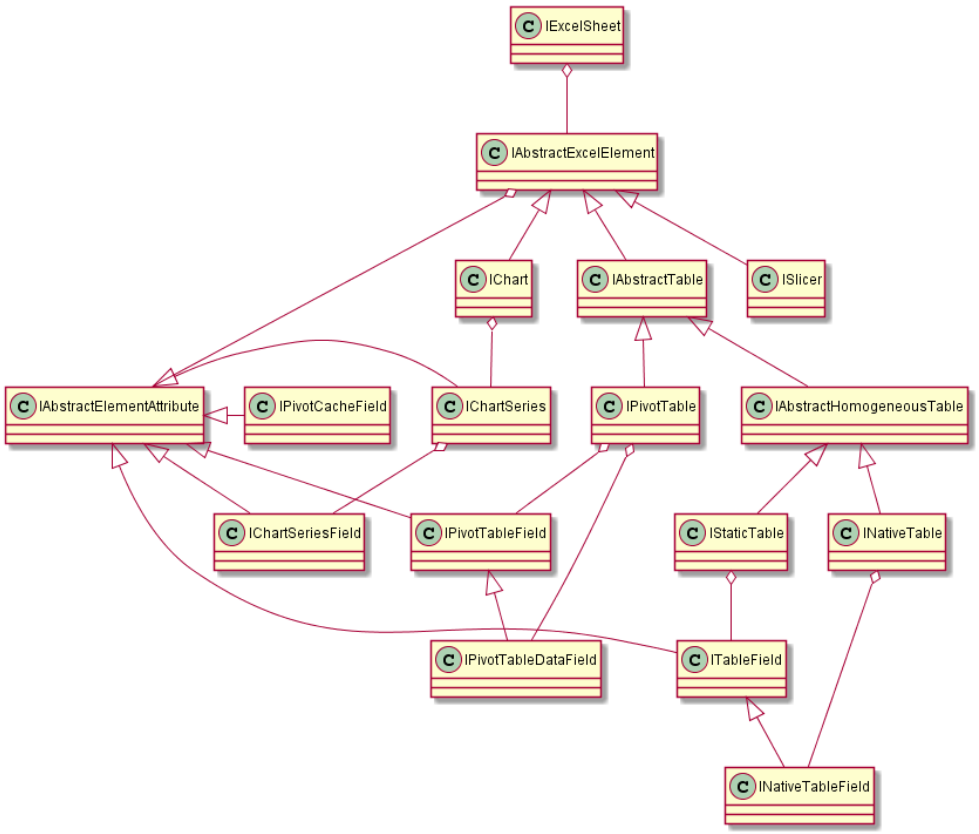
[23] Connection string je řetězec obsahující informace pro připojení k databázi.

[24] JDBC je programové rozhraní pro přístup Java programů do běžně rozšířených databází.

[25] SDK (software development kit) je sada nástrojů určených pro tvorbu a vývoj aplikací.

[26] Tento způsob se označuje jako využití Mashup services.

[27] XPath se používá pro navigaci v XML dokumentech.



Obrázek 3.4: Diagram tříd pro datově zajímavé objekty Excel

obousměrně mapovat na adresářovou strukturu při extrakci specifikací modelů a reportů, aby byl příslušný odkazovaný soubor nalezen.

Pro částečnou analýzu datových toků v Cognos existuje nástroj Cognos Data Lineage, který je integrovaný v Cognos Analytics. Je také značně neaktuální a nezná nové objekty představené v Cognos Analytics 11. Navíc neumí propojit analytické modely s datovými zdroji. Pokrývá pouze části podporovaných objektů a řada toků je díky tomu ztracena. Uz jen prototyp konektoru pro Manta Flow pokryje větší rozsah datových toků v Cognos.

3.5.1 Související nástroje

Cognos má poměrně velký „ekosystém“ podpůrných nástrojů a to zejména na analytické vrstvě. Řada z nich produkuje specifické objekty, které se různě mohou účastnit datových toků. Tyto nástroje s jejich produkty pokrývá následující výčet.

- **Cognos Analytics** – hlavní webová aplikace běžící na Cognos Analytics serveru. Poskytuje grafické uživatelské rozhraní pro tvorbu různých

druhů dokumentů. Je v něm integrován reporting, dashboarding a data modeling.

  - **Reporting** – komplexní nástroj pro tvorbu reportů. Důraz je kladen na volnost, flexibilitu, tabulky a podání široké sady informací. Vychází z Report studia obsaženého v Cognos BI 10. Produktem jsou reporty s XML specifikacemi.
  - **Dashboarding** – intuitivní rozhraní pro rychlou tvorbu dashboardů. Myšlenkou je ušetřit uživateli čas strávený nepodstatnými designovými rozhodnutími, proto je řada stylů a konfigurace předpřipravena. Produktem je *Dashboard*, jehož specifikaci lze z Content store získat ve formátu JSON.
  - **Data modeling** – novinka v Cognos Analytics. Poskytují další úroveň datové abstrakce a umožňují modelovat zdroje dat složené z databází i souborů zároveň. Dokáží také seskupit více modelů do jednoho modulu a obchází tak dřívější omezení, kdy reporty mohly mít jen 1 datový zdroj.

- **Framework manager** – starší, ale stále široce využívaný a vyvíjený analytický modelovací nástroj. Podporuje tabulární i multidimenzionální modely a simuluje tak OLAP funkcionalitu. Jeho produktem jsou *modely* se specifikacemi formátu XML.

- **Transformer** – původní nástroj pro tvorbu OLAP kostek a analytické plánování nad nimi. Už není vyvíjen a v budoucnu bude nejspíš nahrazen nástrojem *Dynamic cube designer*. Hlavním produktem jsou *Power cubes*, které fyzicky kopírují data do jiné lokality na disku. Nástroj má širokou řadu funkcí, ale kapacita jednotlivých kostek je omezena na 2 Gb.[42]

- **Dynamic cube designer** – nejnovější Cognos nástroj pro tvorbu OLAP kostek, zde označených *Dynamic cubes*. Do budoucna mají plně nahradit Power Cubes. Jsou založeny na pokročilém cachovacím systému, který data pouze modeluje dle datového zdroje, ale fyzicky je nikam nepřesouvá. Zaručují výbornou odezvu i při velkém objemu dat, ale jako poměrně nový produkt ještě nepokrývají všechny funkce Transformeru.

3.5.2 Datově relevantní objekty Cognos

V této sekci jsou rozebrány objekty, které lze nalézt ve specifikacích reportů a modelů Framework Manageru a případné závislé objekty, nacházející se v Content store. Není cílem prototypu pokrýt toky ve všech nástrojích a objektech. Vynechány tak například budou *Dashboardy* a závislé objekty a v Cognos Analytics nově představený *Data module*. OLAP analýza bude omezena jen

na dimenzionálně modelovaný FM Model, proto ani *Dynamic cubes* a *Power cubes* nebudou rozebrány.

Report představuje výsledný dokument. Každý objekt tohoto typu má svou vlastní specifikaci, kterou lze extrahovat. V této specifikaci je uveden odkaz na zdroj dat. Specifikace reportu také obsahuje řadu datově zajímavých objektů, které budou popsány dále.

Layout reprezentuje zvolené rozložení stránek. V rámci reportu může být definováno více layoutů, narozdíl od OBIEE je ale viditelný vždy jen jeden zvolený. Tento objekt tedy slouží například k přepínání jazyku prezentace nebo volbě jiného rozvržení stránek apod. Je definován ve specifikaci reportu.

Page (stránka) se nachází uvnitř layoutu, v rámci kterého je její jméno unikátní. Obsahuje prvky určující její rozvržení a hlavně pak datové vizualizace a ovládací prvky, které slouží k interakci s reportem.

Control je uměle zavedený název pro interaktivní prvky, které mění podobu reportu. Jedná se o různé formy záložek, dropdownů[28], tlačítek atp. Některé z těchto prvků mohou využívat dynamická data, proto jsou pro analýzu datových toků také relevantní.

Visualization představuje různé formy grafů, tabulek a vizualizací pro zobrazení dat. Je hodně typů těchto objektů, které mohou mít velmi různou strukturu. Společným rysem je způsob, jakým referencují data. Ve všech těchto vizualizacích se XML atributem *refData* odkazuje na užitý *Query item*.

Query je definováno také ještě ve specifikaci reportu, ale už je součástí spíše logické vrstvy, než prezentační. Slouží k výběru dat z datového zdroje, případně i k provádění dalších kalkulací a filtrací. Query může přímo obsahovat SQL dotazy do databází ve strukturované formě, častěji je ale využíván novější přístup, kdy na analytické vrstvě mezi *Query* a databází je ještě tabulární model nebo OLAP kostka. V tom případě je generován dotaz v MDX syntaxi podle obsažených *Query itemů*.

Query item (report) představuje prvky výsledku dotazu reprezentovaného objektem *Query*. Na *Query itemy* se odkazuje z vizualizací a interaktivních *Control* objektů uvnitř specifikací reportů. Query itemy mohou samy o sobě odkazovat na jiné objekty tohoto typu a to včetně těch, které jsou součástí jiných *Query* objektů.

[28]Dropdown je označení pro prvek uživatelského rozhraní, který vybírá jednu z možností v nabídce.

Model je označení pro produkt nástroje Framework Manager (FM model). Modelují se v něm tabulární i multidimenzionální schémata. V určitém kontextu tedy může představovat OLAP kostku. Je jedním z možných zdrojů dat pro reporty a dashboardy.

Namespace má podobnou funkci jako adresáře v diskových strukturách – funguje jako kontejner pro nezávislá schémata modelu. Jednotlivé namespace bývají modelovány buď tabulárně nebo dimenzionálně. Mohou také obsahovat tzv. *Shortcuts*, což jsou strukturované reference do jiných částí modelu.

Calculation je objekt vypočtený dle jiných prvků v rodičovském *Namespace*.

Query subject je součástí tabulárních *Namespace* a obsahuje SQL dotaz do databáze. V případě, že SQL dotaz neobsahuje, odkazuje vždy na jiný *Query subject*, který tuto podmínku už splňuje.

Query item (model) se pojí ke sloupci databázové tabulky, nebo odkazuje na jiný *Query item*. Zajímavé je, že tento prvek, respektive XML element s tímto názvem, se zde používá i jako potomek dimenzí. Například SSRS pro tento význam používá pojmenování *Attribute*.

Dimension je potomkem multidimenzionálně modelovaného *Namespace*. Podobně jako *Query subject* může obsahovat SQL dotaz nebo referenci na jinou dimenzi, která ho obsahuje.

Hierarchy, Level zde mají význam konzistentní s jejich definicí v OLAP kostce 3.1.1. Společně s dimenzí se nachází ve specifikaci FM modelu.

DataSource označuje datové zdroje, které jsou popsány v rámci samostatných extrahovaných souborů. Tyto soubory obsahují informace potřebné k připojení k datovému zdroji, fyzickou adresu na disku apod.

3.6 Shrnutí

V této sekci shrnu objevené společné rysy analyzovaných reportingových nástrojů, které budou základem pro nalezení společné reprezentace objektů v těchto nástrojích.

I když nebyl tento objekt vždy explicitně uveden, vyskytuje se *Server* reprezentující úložiště reportů v každém z nástrojů kromě Excelu, který si vystačí s jednotlivými reporty uloženými v libovolné adresářové struktuře na disku. Není překvapením, že zásadním pojmem je zmíněný *Report*. V Excelu lze za report považovat *WorkBook* obsažený v samostatném XLS dokumentu.

Prvky reportu jsou vizualizace a různé filtrační nebo kontrolní prvky, které interaktivně s reporty pracují. Tyto prvky byly už v analýze opakovaně

nazývány jako *Report itemy*. Společným znakem Report itemů bylo, že obsahují odkazy na data. Vícekrát se objevil název *Data series* pro datové řady samostatných hodnot. Tyto hodnoty jsou ale napříč nástroji pojmenovány různě a bude pro ně potřeba ve společné reprezentaci zavést jednotné pojmenování.

V Cognos a OBIEE se objevila rozložení reportů nazvaná jako *Layout*, která se dělilá na obsažené stránky (*Page*). V Excelu lze koncept stránky také nalézt pod označením *Sheet*, neboli list. Pouze reporty SSRS jsou jednostránkové a tak danou stránku ani explicitně nezmiňují.

V úvodu kapitoly byly představeny OLAP kostky, které mohou být užity jako zdroje dat ve všech analyzovaných nástrojích. Na dimenzionální model jsem narazil i v průběhu analýzy pro Cognos a nezjistil žádné větší odlišnosti. Při navržení společné reprezentace se tedy dá vycházet z objektů v OLAP kostkách popsaných v sekci 3.1.1.

Jako protipól dimenzionálního modelu byly objeveny napříč nástroji různé podoby tabulárních modelů. Mezi společné objekty nalezené v těchto modelech patří objekty symbolizující potenciálně propojené tabulky, které se skládají z položek, které jsou blízké sloupcům databázových tabulek. Pojmenované byly různě a v návrhu společné reprezentace se přikloním k označení *Table* a *Column*, což se zdá být nejpřesnějším označením významu, přestože se nejedná o fyzickou (databázovou) vrstvu. V modelech se také vícekrát vyskytly různé kalkulace, vypočtené a agregované hodnoty.

Samotné reporty si v SSRS a Cognos vyhradily ještě jednu modelovací vrstvu, která byla součástí reportu. Objekty v ní opět připomínaly tabulky. Na straně SSRS to byly *Data sety* a *Fieldy*, u Cognosu stejná funkce připadla *Query* a *Query item* objektům.

Analýza ukázala, že SSRS, Cognos i OBIEE jsou si poměrně podobné a objekty v jednom nástroji mají často paralelu v ostatních. Pro několik objektů Excelu toto tvrzení platí také, ale jinak je Excel oproti ostatním nejvíce odlišný. To může vycházet z faktu, že Excel se sice používá k reportingovým účelům, ale primárně byl zamýšlen jako tabulkový editor. Některé jeho tabulkové objekty, jako např. *Query table*, nativní a statické tabulky, by teoreticky mohly být namapovány na objekty v ostatních nástrojích, ale rozhodně jde silně o věc interpretace těchto objektů. Společnou reprezentaci tak bude nejlepší řešit primárně na úrovni SSRS, Cognos a OBIEE a Excel řešit samostatně.

Kapitola 4

Návrh společné reprezentace reportingových nástrojů

V předchozí kapitole byly analyzovány datové objekty ve čtyřech různých reportingových nástrojích. Na základě identifikace významově podobných objektů v závěrečném shrnutí je v této kapitole navržen model uzlů pro reportingové nástroje.

Hlavní motivací pro sjednocení uzlů reportingových nástrojů je zjednodušení integrace grafu datových toků do nástrojů třetích stran. V rámci těchto integrací je totiž často nezbytné uzly modelu opět transformovat na jiné objekty specifické pro daný software. Čím větší bude rozmanitost uzlů na straně Manty, tím komplikovanější a nákladnější bude tato transformace.

Podstatným zjištěním, které z analýzy objektů vyplynulo, je skutečnost, že i přes řadu podobných objektů se napříč nástroji vyskytlo velké množství specifických objektů, které se v ostatních nástrojích neobjevily. Tyto unikátní uzly často mají zásadní význam a pokoušet se je uměle mapovat a tento význam měnit jen za účelem sjednocení modelu by bylo nežádoucí.

Možným řešením by bylo významově podobné objekty sjednotit a unikátní objekty jednoduše do modelu přidat s vědomím, že jen malá podmnožina reportovacích nástrojů je využije. Toto řešení jsem zavrhl, protože není škálovatelné. S novými podporovanými reportovacími nástroji by model zbytečně bobtnal a postupně se stal nepřehledným a kontraproduktivním.

Rozumnější variantou je unikátní objekty ze společného modelu vynechat a nechat implementujícímu programátorovi volnou ruku při jejich definici. Místo toho bude lepší zaměřit se na objekty, které se v analyzovaných nástrojích vyskytly opakovaně a z nich vytvořit sadu společných uzlů. Nabízí se dvě varianty, jak poté přistoupit ke zmíněné sadě.

1. **Vynucení využití modelu** – v případě podobnosti uzlu s uzlem společného modelu by bylo preferováno využití již definované varianty. Hrozila by ztráta informace o autentickém pojmenování objektů v daném

nástroji. Tomu by se dalo zabránit zobrazením tohoto názvu v podobě jednoho z atributů uzlu. Tato varianta může trochu brzdit vývoj, pokud se programátor bude příliš rozmýšlet, zda daný uzel má považovat za podobný a využít tak uzel ze společného modelu, nebo zda vytvořit unikátní.

2. **Preference specifických objektů** – tento přístup dává programátorovi volnost v definici objektů a poskytuje pro uživatele jednoduchý přístup k typu objektu. Stále je ale třeba mít o společném modelu povědomí a hlavně být schopen vysvětlit význam uzlů inženýrovi, zabývajícímu se zmíněnými integracemi. V případě potřeby je vyžadována pomoc s mapováním. Určitou nápovědu vyjadřující podobnost vůči uzlu společného modelu by mohlo být například uvedení tohoto protějšku ve formě jeho názvu v atributu vrcholu.

Při implementaci konektoru pro Cognos se přikloním k 2. variantě a dám tak přednost autenticky pojmenovaným typům uzlů, abych předešel jakémukoliv zmatení uživatelů Cognosu spojenému s názvy objektů. Toto rozhodnutí však nebude nezvratné a bude poměrně snadné zpětně zvolit jiný přístup pro kohokoliv, kdo daným objektům rozumí. Stejně tak bude později možné tento přístup změnit i pro ostatní reportingové nástroje, u kterých implementace probíhala ještě v době, kdy společný model nebyl vytvořen nebo finalizován.

Z analýzy objektů dále vyplynulo, že objekty se dají téměř vždy zařadit do jedné ze čtyř skupin. Tyto skupiny jsou navíc uspořádány dle pomyslné vzdálenosti od fyzického zdroje k výslednému reportu. V této práci je proto budu označovat za *vrstvy*. Patří mezi ně:

- **fyzická** – zahrnuje uzly reprezentující datové zdroje a jejich strukturu. Patří do ní například databáze, její schéma, tabulky, sloupce, ale i zdrojové soubory (například v CSV formátu[29]). Analýzu datových toků v této vrstvě ze značné části pokrývá již implementovaná *DataflowQueryService*.

- **analytická** – vrstva, o kterou se často stará dedikovaný nástroj pro tvorbu analytických modelů, OLAP kostek a podobných. Mezi uzly této vrstvy proto patří dimenze, hierarchie, specifické uzly tabulárních modelů apod.

- **logická** – některé reportovací nástroje ještě umožňují další vrstvu abstrakce nad analytickou vrstvou. Typicky se tato vrstva modeluje už v rámci reportovacího nástroje a dané objekty umožňují různé formy kalkulací a agregací nad hotovým modelem.

[29]CSV (Comma-separated values) je formát sloupcově strukturovaných datových souborů.

- **prezentační** – uzly v této vrstvě reprezentují nejčastěji viditelné objekty, které jsou přímou součástí reportů. Případně jsou to objekty, které prezentaci viditelných objektů ovlivňují nebo reprezentují získaná data apod.

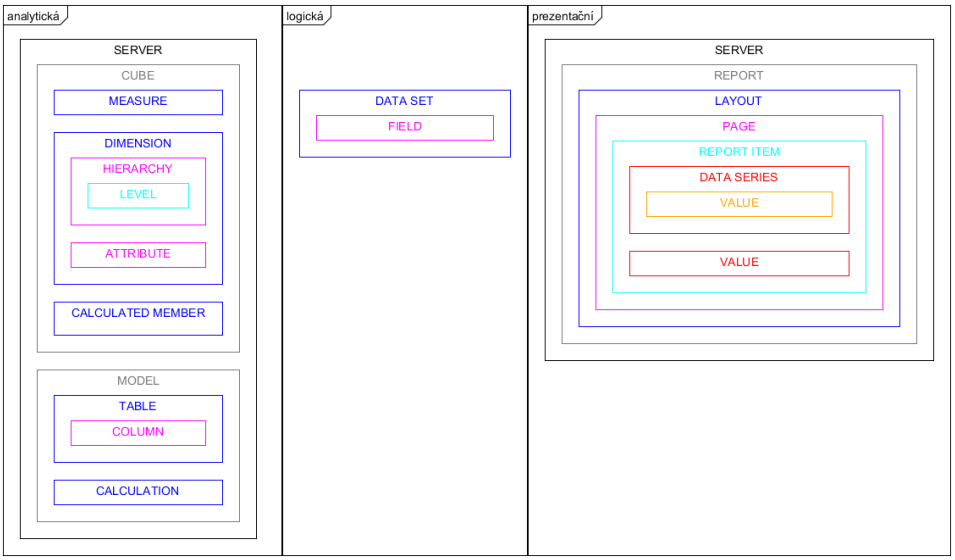
4.1 Model reprezentace objektů

Většina definovaných typů uzlu reprezentuje objekty, které již byly popsány v předchozí kapitole 3 a nebudou proto znovu popsány. Platí, že pokud se stejně pojmenované uzly vyskytují na jiných vrstvách, jedná se o jiné uzly[30]. Při implementaci doporučuji všechny uzly prefixovat názvem jejich vrstvy, aby byly vzájemně odlišeny.

Ve společném modelu uzlů bude vynechána fyzická vrstva, protože databázové uzly i uzly spojené se soubory jako datovými zdroji jsou již v rámci Manta Flow definované a praxí ověřené. Zbytek společných uzlů je seřazen dle vrstvy v následujícím výčtu.

- **analytická**
  - **Server** je již také v kódu definován. Může se vyskytnout ve fyzické, analytické i prezentační vrstvě. Ve všech případech se jedná o jeden stejný definovaný uzel, neboť jeho význam na každé z nich je totožný.
  - **Cube**
  - **Dimension**
  - **Attribute**
  - **Hierarchy**
  - **Level**
  - **Measure**
  - **Calculated member**
  - **Model** – obecné pojmenování pro jakékoliv tabulární modely
  - **Table** – tabulka v tabulárním modelu. Může přímo korespondovat s databázovou tabulkou, nebo být z jedné či více vytvořena.
  - **Column** – sloupec v tabulce tabulárního modelu. Koresponduje se sloupcem databázové tabulky.
  - **Calculation** – obecné pojmenování pro jakékoliv vypočtené hodnoty na základě ostatních členů modelu.

[30] Jedinou výjimkou tohoto pravidla je uzel *Server*, ten není třeba rozlišovat, protože význam je na každé vrstvě stejný.



Obrázek 4.1: Model společných uzlů

- **logická**
  - **Data set** – obecné pojmenování pro objekt připomínající tabulku na logické vrstvě.
  - **Field** – prvek data setu.
- **prezentační**
  - **Server**
  - **Layout**
  - **Page**
  - **Report item** – prvek slučující všechny vizuální prvky na stránce. Zahrnuje grafy, tabulky, listy, atd. Může fungovat i jako kontejner pro řadu hodnot (*Value*).
  - **Data series**
  - **Value** – jednotlivé hodnoty datových řad, případně jiné samostatné hodnoty a proměnné, které report item uchovává.

Pro lepší přehled hierarchických struktur jednotlivých objektů je přiložen diagram 4.1. Úmyslně jsou v něm vynechány vazby mezi jednotlivými objekty pro lepší přehlednost.

Cognos objekt	Vrstva	Uzel společného modelu
Report	prezentační	Report
Layout	prezentační	Layout
Page	prezentační	Page
Control	prezentační	Report item
Visualization	prezentační	Report item
Query	logická	Data set
Query item (report)	logická	Field
Model	analytická	Model
Namespace	analytická	x
Calculation	analytická	Calculation
Query subject	analytická	Table (a.v.)
Query item (model)	analytická	Column (a.v.)
Hierarchy	analytická	Hierarchy
Dimension	analytická	Dimension
Level	analytická	Level

Tabulka 4.1: Mapování objektů Cognos na společný model

4.1.1 Příklad mapování objektů na uzly pro Cognos

Pro demonstraci, jak by mohlo vypadat převedení specifických objektů reportingových nástrojů na společný model uvádím příklad mapování objektů IBM Cognos. Z tabulky 4.1 je zřejmé, že se povedlo namapovat všechny uzly krom jediného. Objekt *Namespace* se totiž nevyskytl v žádném z ostatních nástrojů ve stejném významu a proto nebyl definován do společného modelu.

Kapitola 5

Návrh rozhraní modelu pro Cognos

Jedním z hlavních bodů návrhu konektoru pro Cognos, který ovlivní téměř celou implementaci je návrh tříd reprezentujících objekty, které lze nalézt uvnitř extrahovaných specifikací. Dle konvence v ostatních konektorech Manty budou implementace všech tříd odděleny od jejich rozhraní. Tato kapitola popisuje právě návrh daných rozhraní.

Analýza odhalila, že většina objektů ve specifikacích nástroje Cognos má značně heterogenní povahu, která bude vyžadovat specifický přístup zejména k získávání potomkovských objektů a řešení odkazů na jiné objekty, které se vyskytují pod mnoha různými XML značkami. Kvůli tomu bude téměř každý objekt ve specifikaci, který je zajímavý z hlediska datových toků, nebo jeho rodič zajišťující jeho unikátnost v hierarchii, mít vlastní třídu a chování.

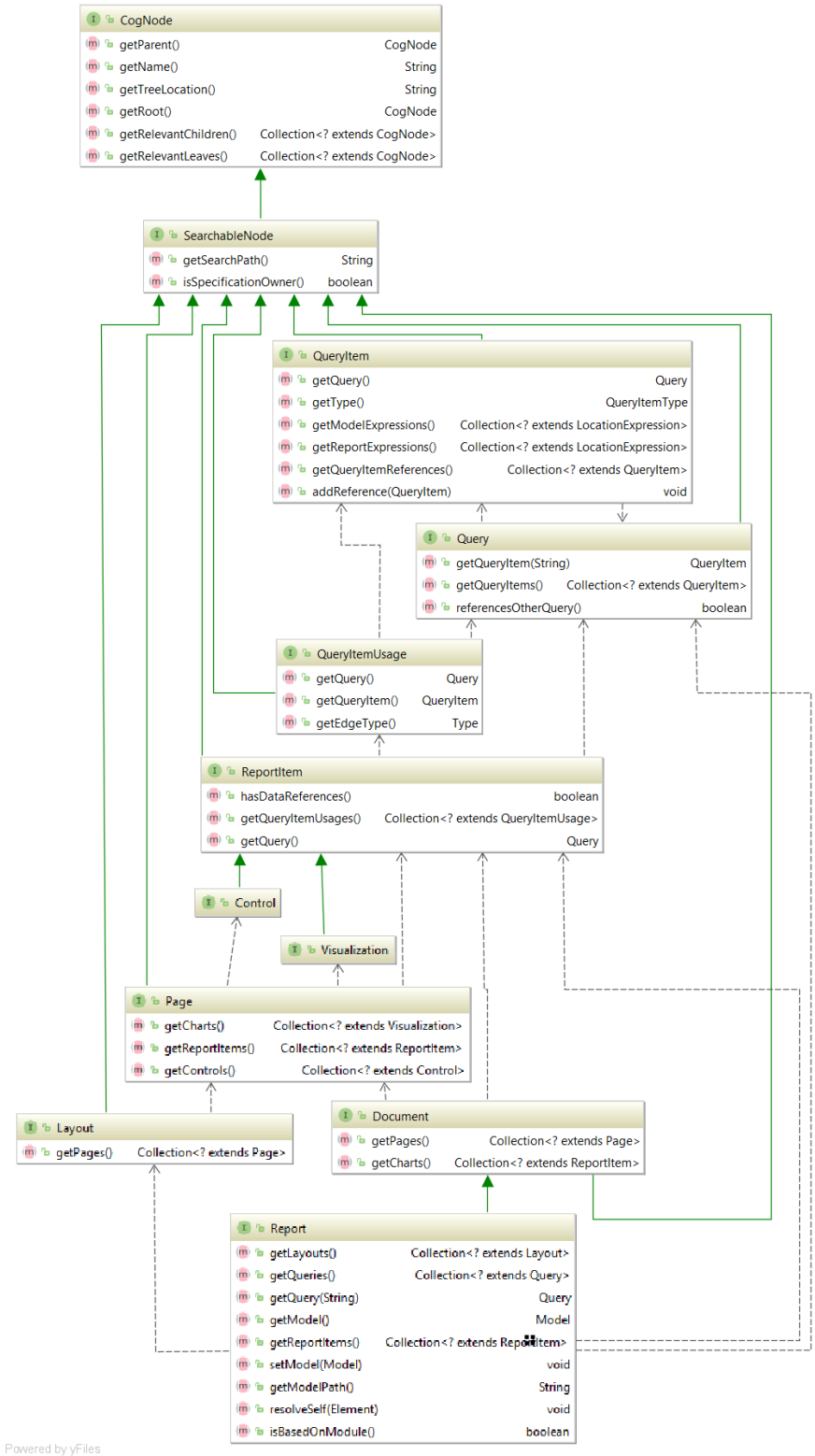
V této a dalších kapitolách budou použity diagramy, které většinou budou respektovat notaci UML. Některé z nich se ale od notace odchýlí.

5.1 Diagram rozhraní

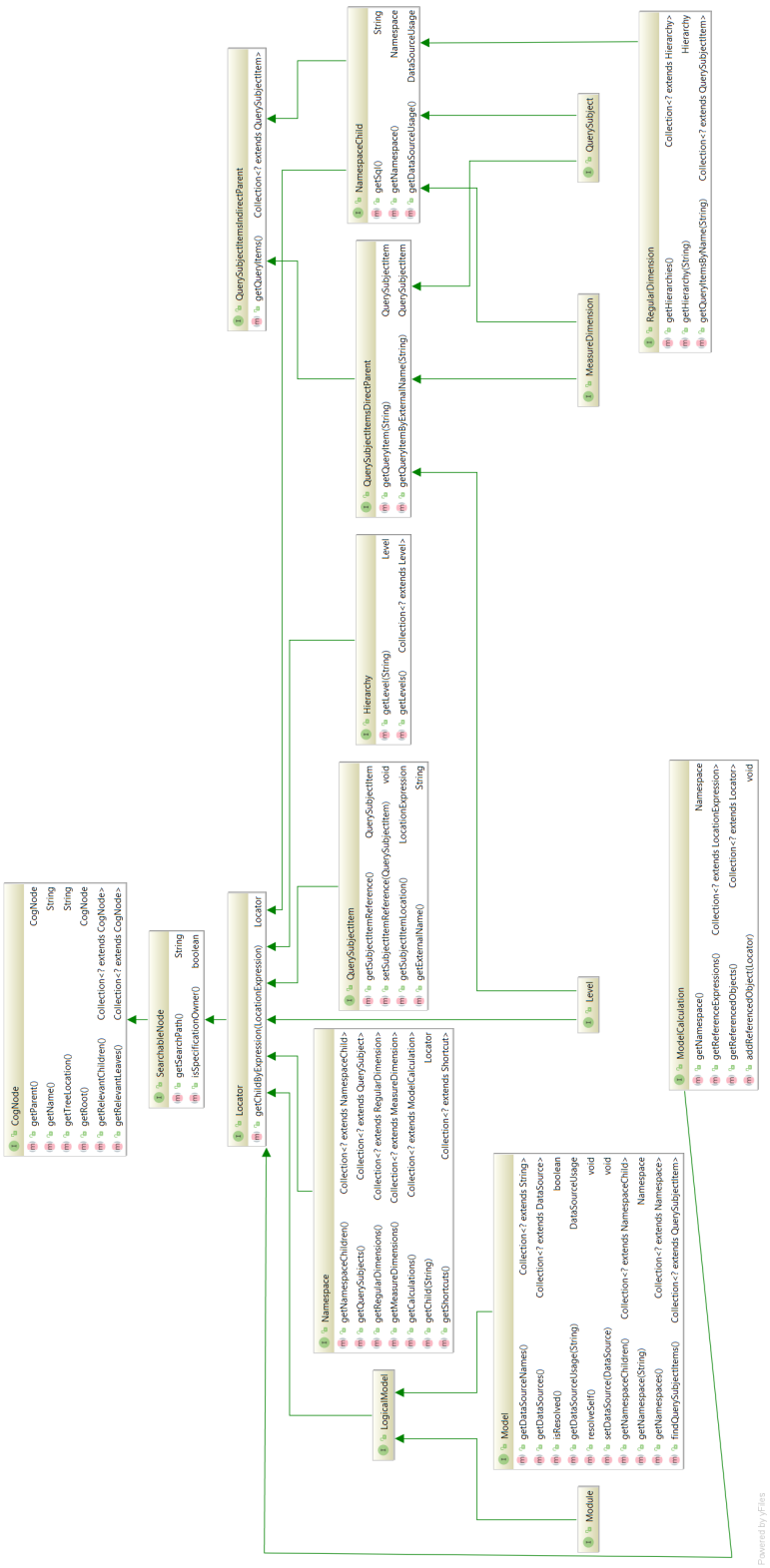
Celkový počet navržených rozhraní převyšuje 30. Tento počet rozhraní není rozumné zobrazit na jednom diagramu. Proto jsem model rozdělil do diagramu popisujícího rozhraní převážně závislých na reportu (obr. 5.1) a souvisejících s FM modelem (obr. 5.2).

V následujícím seznamu jsou rozhraní, u kterých má jejich popis přidanou hodnotu a ta, která v diagramech nebyla ukázána z důvodu lepší přehlednosti.

CogNode je společný předek pro všechna rozhraní, která jsou součástí hierarchií a transformovaná na uzel se vyskytnou ve výsledném grafu. Slučuje typické funkce stromových struktur, jako jsou metody *getParent()*, *getRoot()* a podobné.



Obrázek 5.1: Diagram rozhraní prezentační a logické vrstvy



Obrázek 5.2: Diagram rozhraní analytické vrstvy

DataFlowInput představuje nezávislou jednotku, která je vstupem pro scénář generující datové toky. Seskupuje analyzovaný dokument, zdrojové modely a informace o fyzických datových zdrojích.

DataSourceUsage reprezentuje datový zdroj v kontextu užití FM modelem. Řeší se zde aliasy přiřazené datovému zdroji v modelu, použité databázové schéma apod.

Document je společné rozhraní pro výsledné produkty reportingu, tj. především reporty, ale i dashboardy. Prozatím slouží k přístupu ke stránkám a report itemům.

LocationExpression poskytuje metody pro práci s výrazy, které slouží jako lokalizátor určitého objektu v rámci hierarchické struktury reportů a modelů.

Locator indikuje, že daný objekt splňující toto rozhraní, umí zpracovávat *LocationExpression*, konkrétně dokáže vrátit potomka metodou *getChildByExpression()*.

LogicalModel seskupuje společné vlastnosti modelů analytické vrstvy, které jsou zdroji dat pro výsledné dokumenty. Toto rozhraní rozšiřují např. *Model* a *Module*.

NamespaceChild je značení pro potomka *Namespace*, který zároveň nese informaci o použitém SQL dotazu nad databází. Toto rozhraní tak seskupuje společné vlastnosti *QuerySubject* a dimenzí.

QueryItemUsage reprezentuje *QueryItem* v kontextu užití některou z vizualizací v reportu. Dodává informace o způsobu užití potřebném pro identifikaci typu hrany v grafu datových toků.

QuerySubjectItemsIndirectParent slouží pro indikaci, že uvnitř objektů splňujících toto rozhraní jsou adresovatelné *QuerySubjectItemy*.

QuerySubjectItemsDirectParent je kontejner rozšiřující *QuerySubjectItemsIndirectParent* o metody typické pro objekty přímo obsahující potomky typu *QuerySubjectItem*.

ReportItem seskupuje společné vlastnosti objektů umístěných na stránce reportu. Jedná se o předka pro rozhraní *Control* a *Visualization*.

SearchableNode je další z rozhraní, která jsou předkem pro velkou většinu ostatních. Přímo rozšiřuje *CogNode* a dodává informace o searchpath uvnitř Content Store a metody potřebné pro práci s ní.

RegularDimension modeluje standardní dimenzi, která obsahuje hierarchie, levely atd.

MeasureDimension představuje dimenzi, která přímo obsahuje jednotlivá *Measures*. Ty mají ve specifikacích ale totožnou podobu jako *Query item*, proto není nutné vyčleňovat pro ně speciální rozhraní.

QuerySubjectItem reprezentuje objekt *Query Item* na analytické vrstvě. Pojmenování *QuerySubjectItem* je zde použito právě pro odlišení těchto objektů, které se v Cognos vyskytují na logické i analytické vrstvě.

Dále pouze uvádím pouze výčet těch rozhraní, jejichž význam je zřejmý, protože reprezentují objekty již popsané v sekci 3.5.2.

- **Control**
- **DataSource**
- **Hierarchy**
- **Level**
- **Model**
- **ModelCalculation**
- **Namespace**
- **Page**
- **Query**
- **QueryItem**
- **QuerySubject**
- **Shortcut**
- **Visualization**

5.2 Interakce tříd, orchestrace

Pro lepší pochopení způsobu, jakým spolu budou komunikovat a spolupracovat třídy při řešení méně přehledných problémů, jsem se rozhodl navrhnout tuto interakci formou sekvenčních diagramů. Ty pomohou při implementaci a zároveň mohou sloužit jako určitá forma dokumentace, neboť většinou poskytují jednodušší a od implementačních detailů odproštěný přehled o dění v programu během řešení méně přehledných scénářů.

V diagramech záměrně vynechávám některé metody, které pravděpodobně budou v kódu, ale nejsou pro pochopení scénáře podstatné, zjednodušuji smyčky a podmínky. Nebudou také uvedeny zpětné šipky symbolizující návrat

volání kvůli lepší přehlednosti diagramů. Zodpovědnost za zpracování přijaté zprávy v každém časovém okamžiku lze i tak pochopit ze žlutě označených polí symbolizující aktivitu třídy.

5.2.1 Průběh extrakce

Návrh komunikace tříd při extrakci potřebných dat z Cognos Content Store je uveden na diagramu 5.3. Je patrné, že zodpovědnost za iniciaci procesu mají *ExtractorScenario* a posléze *ExtractorTask*. Obrázek popisuje úspěšný případ, kdy je navázáno spojení do úložiště Cognos objektů a je zahájena extrakce reportů.

Za použití Cognos SDK třídy *ContentManagerService* je získána třída reprezentující report na straně Cognosu. Následuje uložení na disk svěřené třídě *BaseClassSaver*, která vytvoří pomocný *CogReport* objekt, jenž obsahuje logiku získání specifikace reportu, která je následně na disk zapsána ve formě XML dokumentu. Scénář končí uvolněním zdrojů, mj. přerušením spojení s Content Store.

5.2.2 Zpracování extrahovaného reportu

Po úspěšné extrakci jsou uloženy na disku XML specifikace reportů a objektů analytické vrstvy. Sekvenční diagram 5.4 ukazuje, co se s těmito daty děje dál.

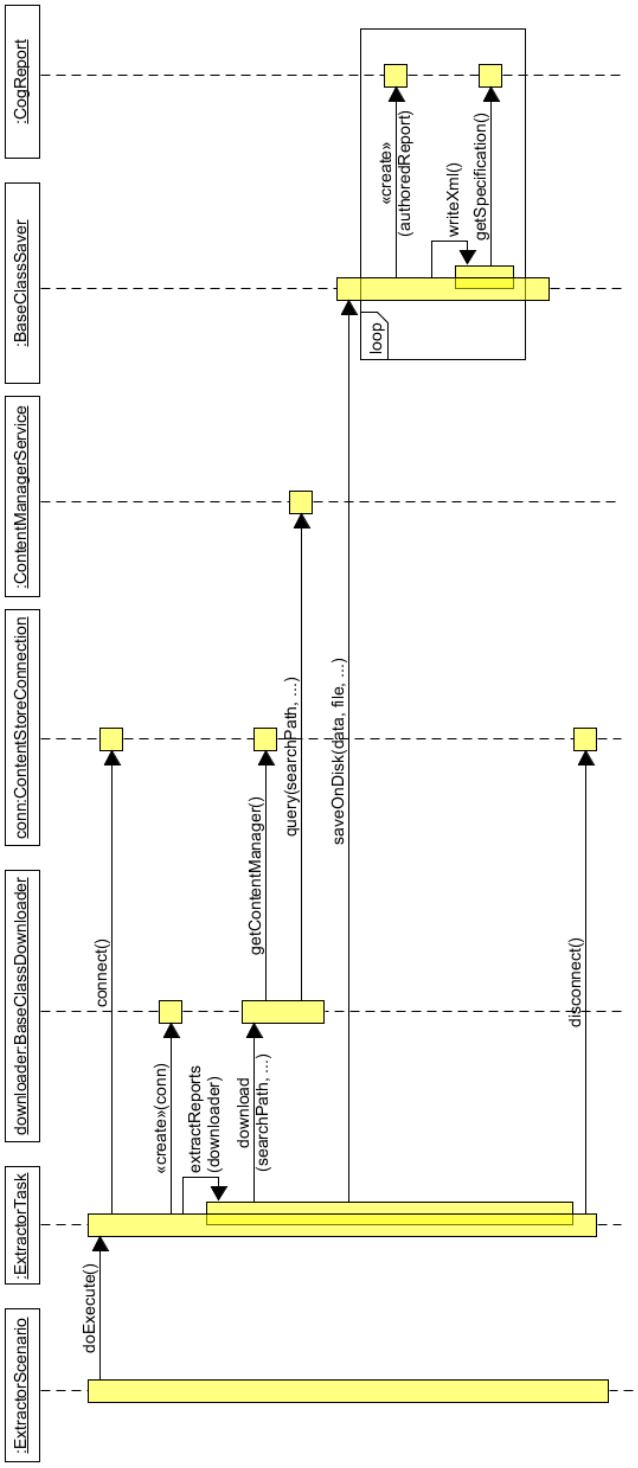
Za načtení souboru reportu je zodpovědná třída *CogInputReader*. Ta na základě specifikace reportu iniciuje částečný resolving třídy *Report* tak, aby vytvořený objekt následně poskytl informace o potřebném FM modelu. Reader poté požaduje po *ProviderCache* vytvořený objekt *Model*. Pokud ještě neexistuje, cache ho vytvoří a předá k dalšímu zpracování.

Obdobným způsobem jsou poté získány datové zdroje relevantní pro *Model*, pro který je voláním metody *resolveSelf()* iniciován celý resolving. Resolvovaný model je prerekvizitou pro resolving reportu, proto pak *CogInputReader* zašle zprávu reportu, že je třeba vykonat zbývající resolving všech potomkovských objektů.

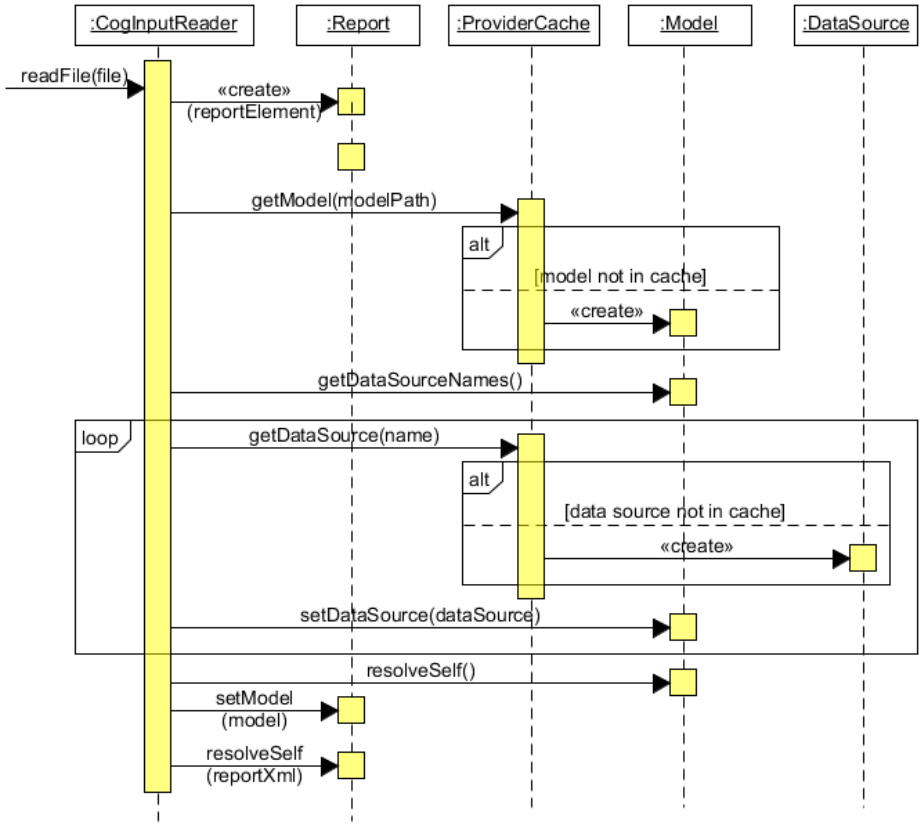
5.2.3 Průběh generování grafu datových toků

Ve chvíli, kdy jsou report i objekty, na kterých závisí, resolvované, může přijít na řadu generování grafu datových toků. Tento proces je pro jeden vstup demonstrován diagramem 5.5. Hierarchická struktura objektů pod reportem a modelem bude pravděpodobně značně komplexní a na místě ilustrovaných smyček bude v implementaci s velkou pravděpodobností větší množství smyček vnořených.

Obdobně jako u extrakce zde má roli iniciátora scénář, v tomto případě *CognosScenario*, který daný vstup předá ke zpracování třídě *CognosDataflowTask*. Ta nejprve deleguje generování uzlů na základě resolvovaného mo-



Obrázek 5.3: Sekvenční diagram orchestrace generátoru datových toků

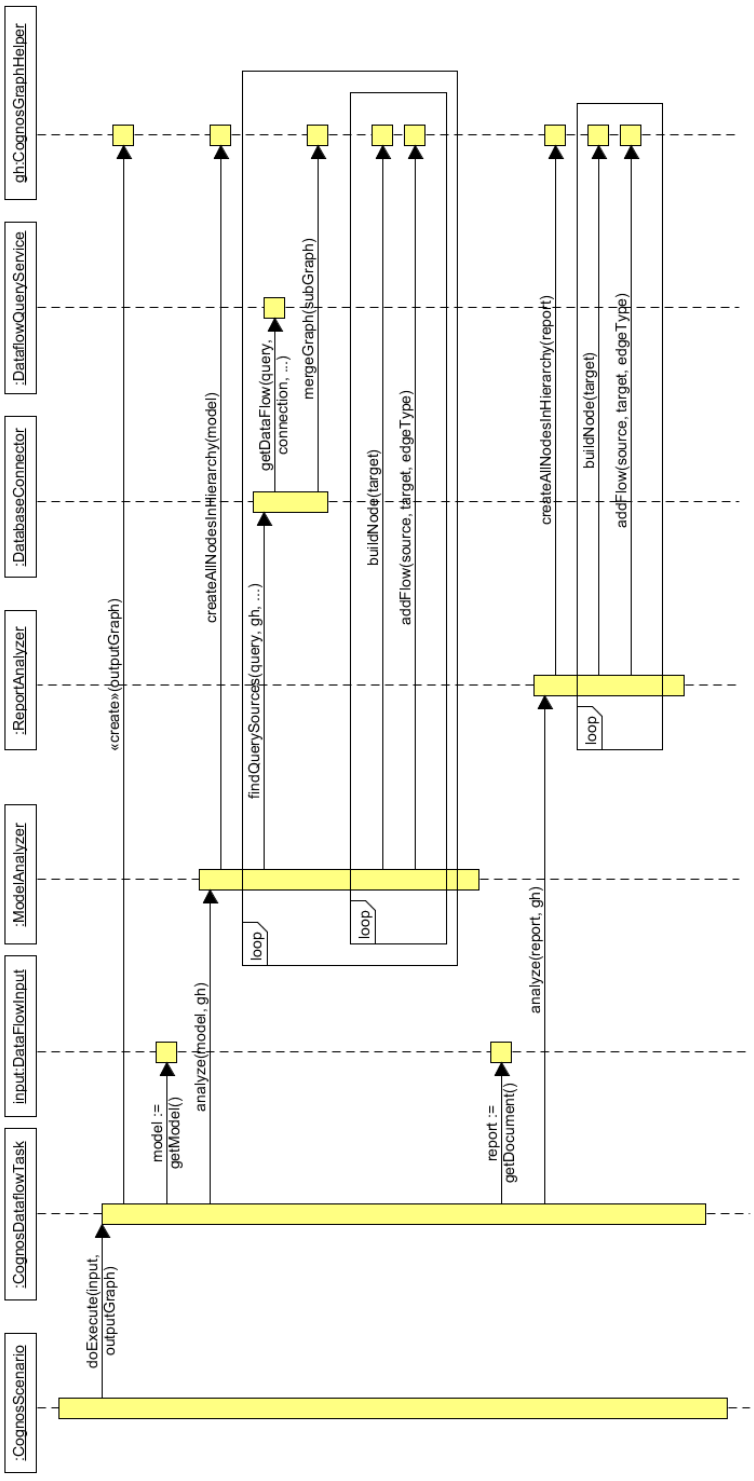


Obrázek 5.4: Sekvenční diagram zpracování extrahovaného reportu

delu třídě *ModelAnalyzer*. Tento objekt nejprve vytvoří uzly grafu pro všechny objekty hierarchie pod kořenovou třídou *Model*.

Pro každý objevený SQL dotaz je posléze využit *DatabaseConnector* a *DataflowQueryService* k získání relevantních uzlů reprezentujících fyzickou databázovou vrstvu, tedy tabulky, sloupce apod. Hranou datového toku jsou tyto uzly spojeny s již existujícími uzly reprezentující objekty v *Modelu*.

ReportAnalyzer má poté podobnou úlohu jako *ModelAnalyzer* s tím rozdílem, že neřeší SQL dotazy. Naopak ale musí propojit některé vnitřní objekty s koncovými objekty v modelu a spojit tak datovým tokem report s modelem. Celý popsaný proces je v rámci *CognosScenario* zopakován pro všechny načtené reporty.



Obrázek 5.5: Sekvenční diagram orchestrace generátoru datových toků

Kapitola 6

Implementace

6.1 Použité technologie a knihovny

Při implementaci konektoru pro Cognos bylo použita řada technologií a externích knihoven. Výběr mnoha z nich byl mimo mou kompetenci, neboť byla požadována konzistence se sadou nástrojů, kterou Manta využívá, a dodržení určitých konvencí. U všech externích knihoven, které byly již v jiném modulu Manty využity, je také nutnost použít je ve stejné verzi, aby nedocházelo při překladu a sestavení celého CLI ke kolizím a nejednoznačnostem. Níže je uveden výběr klíčových nástrojů, které byly použity.

- **Java 8** – podle TIOBE indexu stále ještě nejvyhledávanější programovací jazyk, který je v Mantě zaveden již od začátku.[43] Manta Flow je nezávislý na cílové platformě právě díky virtuálnímu stroji Java Virtual Machine, který interpretuje bytecode[31] a je k dispozici pro všechny běžné operační systémy.
- **JUnit** – nejrozšířenější framework pro testy[32] v Java.
- **Cognos SDK** – sada webových služeb, knihoven a programových rozhraní určených k programatickému ovládání většiny funkcí, které umožňuje nástroj Cognos Analytics. Cognos SDK podporuje automatizovanou správu a implementaci vlastních rozšíření nad reportingovým systémem. [45] V rámci konektoru pro Cognos je SDK použito pro extrakci dat z interního úložiště.
- **interní Manta knihovny** – konektor pro Cognos se opírá o řadu již implementovaných modulů. Zásadní je využití dataflow-generator-

[31] Bytecode je množina instrukcí pro Java Virtual Machine, který ho dokáže interpretovat a vytvořit z něj strojový kód pro specifický operační systém. [44]

[32] Jednotkové (unit) testy slouží k otestování malých částí aplikace nezávisle na zbytku celku.

common-query pro získání uzlů reprezentujících fyzickou vrstvu. Pro načítání a zapsání XML souborů byl dále užitečný connector-xml a třídy spojené s grafem datových toků poskytl dataflow-model.

- **dom4j** – knihovna pro práci s XML strukturami.
- **Spring** – framework pro Java s širokým využitím. Zde je použit pro zavedení aplikace, vytvoření a konfiguraci potřebných tříd formou JavaBean[33].
- **Maven** – správa a automatizace sestavení aplikace (tzv. buildů).
- **Subversion (SVN)** – systém pro správu verzí kódu.
- **IntelliJ IDEA** – vývojové prostředí s vhodnými integracemi pro Java, Maven, Subversion a mj. i pluginem pro automatické generování diagramů tříd.

6.2 Balíčky

Při rozhodování ohledně členění kódu do jednotlivých balíčků i jejich pojmenování jsem z velké části ctil zaběhnuté konvence firmy Manta a struktury podobných konektorů, aby druhá osoba členění kódu snáze rozuměla. Graf závislostí mezi balíčky je vidět na diagramu 6.1. Nejsou v něm záměrně pro přehlednost zobrazeny závislosti na externí knihovny ani jiné moduly Manty.

connector je jedním z balíčků, jehož samostatné vyčlenění bylo sporné. Momentálně obsahuje jen 2 třídy – *CogInputReader* a *ProviderCache*. Pozitivním přínosem je snížené provázání, neboť dataflow-generator díky osamostatnění tohoto balíčků nezávisí na extraktoru ani resolveru. Balíček řeší načtení potřebných vstupů z extraktoru pro následnou analýzu.

connector-extractor řeší problematiku připojení ke Content Store, stažení tříd Cognos SDK reprezentujících extrahované objekty, jejich případné načtení do struktury dle použitého formátu a konečné uložení na disk s případnými dodatečnými metadaty, která už si tvoří extraktor sám.

extensions obsahuje třídy reprezentující extrahované objekty. Tyto třídy jsou závislé na jejich obdobách v Cognos SDK knihovně, ale mají pouze funkce relevantní pro získání specifikací a jejich čištění[34].

[33] JavaBean je speciální typ Java třídy, kterou lze použít mnoha způsoby k zjednodušení vývoje programu. [46]

[34] Čištěním se myslí odstranění nepotřebných informací a případné jednoduché transformace těch potřebných.

Z tohoto pohledu tak tyto třídy fungují dle návrhového vzoru Fasáda,[35] jen kromě zjednodušení rozhraní ještě provádí dodatečné operace.

connector-model obsahuje pouze sadu Java rozhraní. Všechny objekty, které se zapojují do datových toků, zde mají svou podobu. Kromě rozhraní se zde nalézá několik výčtových typů (enum), které většinou slouží pro bližší určení typu daného objektu v případech, kdy by případné řešení dědičností zbytečně komplikovalo model.

connector-resolver je v podstatě implementací modelu. Obsahuje převážně třídy, které se jmenují jako třídy v modelu a navíc mají suffix *Impl*. Zodpovědností těchto tříd je většinou schopnost z relevantního kusu specifikace načíst potřebné informace vztahující se k dané třídě a zároveň iniciovat resolving v potomkovských objektech, jsou-li nějaké. Důležitým úkolem je také vyřešení referencí mezi danými objekty. Tyto reference většinou vedou právě na datový tok.

dataflow-generator jako celek je zodpovědný za transformaci získaného (resolvovaného) modelu na strukturu Grafu datových toků. Přímo obsahuje pouze dvě třídy – *CognosScenario* a *CognosDataflowTask*, které generování datového toku pro poskytnutý vstup řídí.

analyzers je seskupením tříd, které se specializují na generování relevantních uzlů a hran datových toků v specifických hierarchiích objektů. Momentálně jsou zde např. *ReportAnalyzer* a *ModelAnalyzer*, do budoucna přibude rozhodně *DashboardAnalyzer* atd.

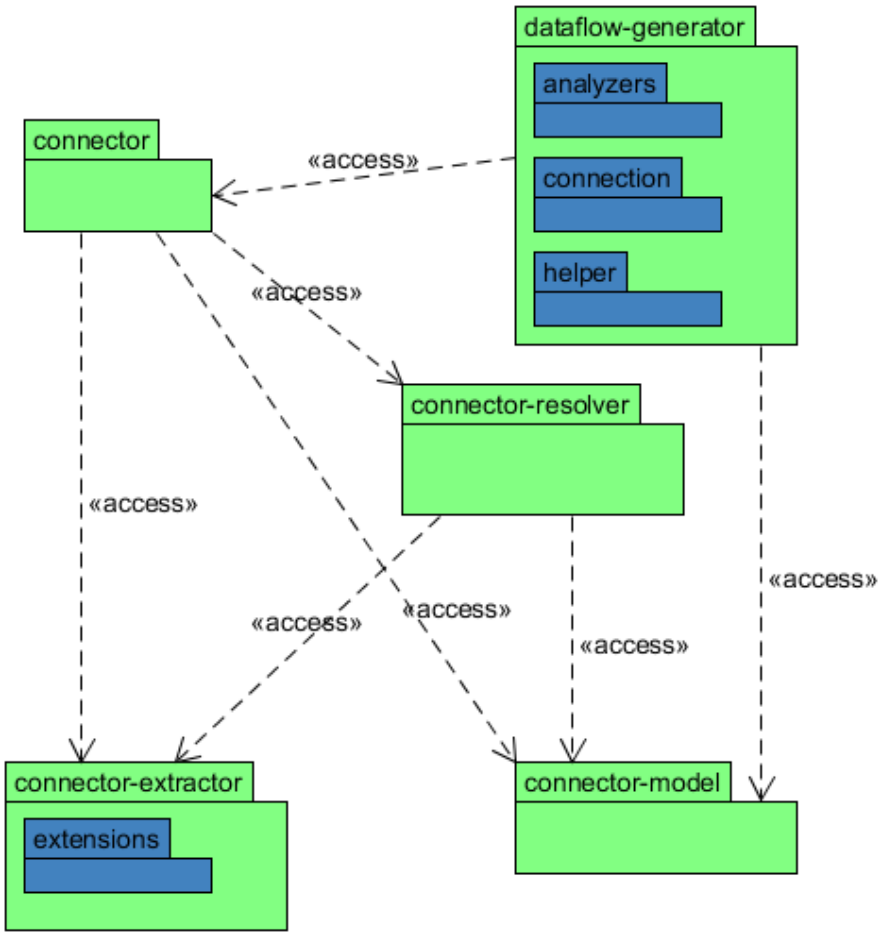
connection obsahuje třídy řešící napojení na fyzické zdroje dat. Nalezneme zde mj. *DatabaseConnector*.

helper v sobě ukrývá třídu *CognosGraphHelper*, která je zodpovědná za všechny operace s grafem datových toků, tedy hlavně za přidávání uzlů a hran.

Kromě zmíněných balíčků, které jsem sám vytvářel, vyžadovala implementace ještě zásahy do již existujících modulů.

- **manta-dataflow-model** – bylo třeba dodat vlastní typy uzlů pro technologii Cognos umístěné ve třídě *NodeType*. Zároveň probíhala už během vývoje snaha o částečné sjednocení některých již definovaných typů uzlů podle modelu navrženého v kapitole 4.1.
- **manta-dataflow-cli** – jednalo se o dopsání několika konfiguračních souborů, souborů pro zavedení programu a úpravu spouštěcích skriptů Manta Flow CLI.

[35] Fasáda je softwarový návrhový vzor sloužící k zjednodušení složitých systémů. Typicky nabízí jen podmnožinu metod.



Obrázek 6.1: Diagram balíčků

6.3 Vybrané řešené problémy

Během pokročilé fáze návrhu a implementace vyvstalo několik problémů. Některé z nich bylo třeba vyřešit ideálně ještě designovým rozhodnutím, jiné přinesly nutnost refactoringu. Spíše než popisování troubleshootingu[36] zde uvádím výčet těch zajímavějších problémů:

- **Podpora různých verzí Cognos** – jedním z požadavků byla podpora verzí Cognos 10[37] a novější Cognos 11[38]. Bylo třeba identifikovat rozdíly mezi jednotlivými verzemi, a to především „breaking chan-

[36] Troubleshooting je označení pro hledání příčiny chyb a jejich opravu.
[37] Cognos 10 se ještě nazýval Business Intelligence Suite.
[38] Cognos 11 už má označení Analytics.

ges“[39]. Největšími změnami ve verzi 11 bylo přidání modulů jako datových zdrojů, tvorba dashboardů jako alternativy k reportům a pár drobnějších změn ve specifikaci reportu. V konfiguraci extraktoru proto nabízím uživateli zvolit správnou verzi, následkem čehož se stáhnou jen vhodné objekty. Při práci se specifikací reportů v resolveru pak počítám s oběma variantami.

- **Potenciálně obrovské úložiště reportů** – je třeba brát v potaz, že zákazník může pustit analýzu toků nad celým úložištěm dokumentů, které může mít tisíce reportů, stovky modelů apod. Pokud by se program pokusil specifikace všech těchto objektů v jednu chvíli načíst do paměti, došlo by k násilnému ukončení pro její nedostatek. Tento problém má řešit třída *DataFlowInput*, která seskupuje vždy jen jeden dokument, modely, které jako zdroje dat využívá, a informace k datovým zdrojům (např. databázím), které naopak využívají modely. Protože report v naprosté většině případů využívá jen jeden nebo několik málo modelů, do paměti se nikdy nemusí načítat více dat, než je nutné. Zmíněná třída je brána jako vstup pro *CognosScenario*[40], která vždy pracuje jen s jednou její instancí zároveň.

- **Redundance resolvingu modelů** – analýza datových toků považuje za vstup jeden unikátní report. Ty ale mohou odkazovat na modely, které už byly v rámci analýzy jiného reportu dříve zpracovány. Proto jsem vytvořil synchronizovanou třídu *ProviderCache*, která ukládá limitované množství resolvovaných modelů a slouží jako jejich poskytovatel. V případě, že model je již uložený v cache, jeho resolving se neopakuje. Podobným způsobem byla rozšířena stejná funkčnost i na datové zdroje, které jsou odkazovány právě z modelů.

- **Redundance generování dataflow pro modely** – podobně jako u resolvingu, i u generování dataflow může docházet z podobných důvodů k opakovanému generování uzlů grafu reprezentujícímu stejný model. Tento problém není tak zásadní, neboť jeho řešení je jen mírná optimalizace, proto v prototypu ještě není řešen, nicméně návrh řešení je připraven. Do třídy *Model* se přidá informace o proběhlém generování datových toků a před spuštěním analýzy modelu se tato informace adekvátně vyhodnotí.

- **Vzájemné reference mezi sourozeneckými objekty** – může nastat situace, že jednotlivé query itemy uvnitř reportů a modelů na sebe vzájemně odkazují. To by mohlo způsobit problémy během resolvingu,

[39] Breaking changes je označení pro změny v nových verzích programu, které narušují zpětnou kompatibilitu.

[40] CognosScenario je třída zodpovědná za spuštění generování datových toků pro daný vstup.

pokud bych danou referenci chtěl řešit v místě její detekce, protože referencovaný objekt ještě nemusí v programu existovat. Proto v těchto případech ukládám pouze informaci o jménu cílového objektu. Teprve až jsou resolvovány všechny objekty stejného typu (sourozenci), prochází kód uložené informace o referencích a dané objekty adekvátně propojí.

- **Zakódované specifikace modulů** – jeden z problémů, který nevyžadoval designové řešení. Spíše jen zbrzdil vývoj. Specifikace modulů jsou totiž zakódovány v kódování Base64,[41] se kterým jsem se předtím nesetkal, a tak jsem měl problém ho identifikovat. Situaci komplikoval fakt, že po dekódování se zobrazila binární data formátu zip. Tento problém nakonec vyřešilo pár řádek kódu.

6.4 Funkce prototypu

Jedním z cílů práce bylo vytvořit funkční prototypy konektoru a generátoru toků pro Cognos. Prototyp je v tomto případě klíčovým slovem, protože pokrýt všechny funkčnosti a objekty, se kterými se v Cognos pracuje, by bylo výrazně nad časový rámec diplomové práce.

Během implementace jsem kladl důraz na vytvoření dostatečně detailní analýzy pro určenou podmnožinu zmíněných funkčností. Držel jsem se myšlenky, že je nutné vytvořit kompletní kostru řešení, která bude všechny objevené funkčnosti brát v potaz pro pozdější implementaci, a zároveň ty významné a u zákazníků nejpoužívanější implementovat. Výčet zásadních implementovaných funkčností je uveden níže.

- **Extrakce specifikací** – v současné chvíli umí extraktor získat a na disk uložit specifikace pro:
  - reporty
  - dashboardy
  - FM modely
  - datové moduly
  - informace o databázových datových zdrojích
- **Resolving objektů** – ze získaných specifikací se relevantní metadata transformují na Java model navržený v kapitole 5.1. Podporovány jsou momentálně všechny relevantní datové objekty obsažené ve specifikacích
  - reportů a interaktivních reportů,
  - FM modelů,

[41] Base64 je kódování, které převádí binární data na tisknutelné znaky.

  - databázových datových zdrojích.

- **Generování datových toků** – pro všechny zpracované objekty ve fázi resolvingu prototyp umí generovat datové toky. Pár nedokonalostí v pokrytí je popsáno v sekci 6.5.

- **Začlenění do Manta Flow CLI** – řešení je integrováno do modulu CLI včetně definice potřebných typů uzlů, úpravy skriptů spouštějících extrakci i analýzu a konfiguračních souborů pro framework Spring.

- **Testování a dokumentace všech modulů** – pro každý z balíčků byla napsána řada JUnit testů, které ověřuji korektní funkčnost často s využitím ukázkových reportů a modelů a snaží se bránit regresím při další implementaci. Celý kód je na úrovni *public* metod, tříd, rozhraní a členských proměnných zároveň poctivě komentován za použití technologie Javadoc[42] a příslušného formátu.

Na obrázku 6.2 je zobrazena vizualizace konkrétního datového toku pro vybraný line graph[43] nacházející se v reportu. Fialově barvené uzly představují databázové zdroje. Nad těmito zdroji je vytvořený poměrně komplexní dimenzionálně modelovaný FM model, proto je patrné množství vnitřních referencí uvnitř modelu.

Vizualizace pro Cognos ještě musí projít konfigurací, která pomůže sloučit panely tak, aby se v ní neobjevoval stejný uzel vícekrát. Doplněna bude ještě i sada ikon pro jednotlivé objekty technologie Cognos a barevné téma.

6.5 Plán rozšíření

V předcházející sekci byl vysvětlen význam a rozsah prototypu a současný stav implementace. Dále je však třeba mít přehled o funkčnostech, které je třeba implementovat, aby pokrytí datových toků bylo co nejpřesnější. Do budoucna je proto v plánu konektor rozšířit o následující položky:

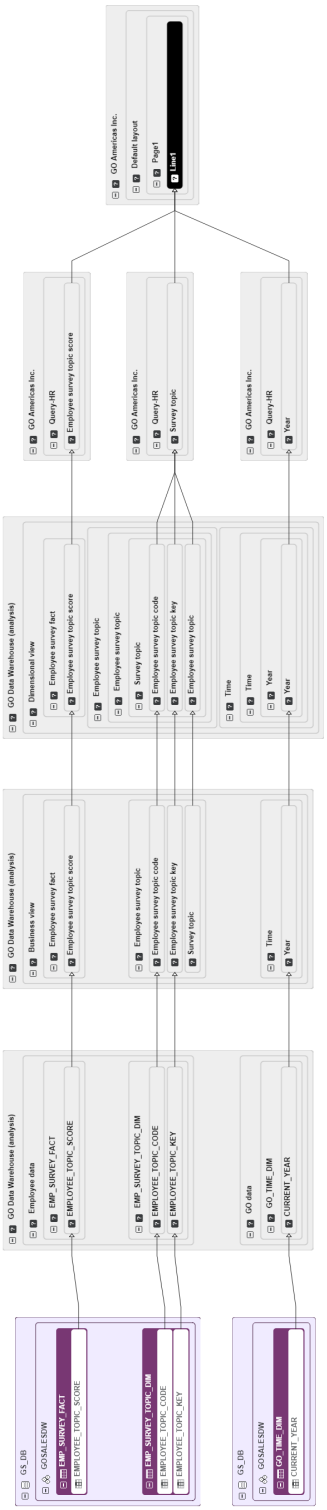
- **Pokrytí modulů** – v Cognos Analytics[44] je nutné jakékoliv zdroje dat pro report zapouzdřit do datového modulu. Současná implementace však pokrývá jen typický způsob dodání dat, a to pomocí Framework Manager modelů. Pro verzi 11 je tak třeba vyřešit začlenění FM modelu do datového modulu, stejně jako začlenění ostatních datových zdrojů[45].

[42] Javadoc je generátor dokumentace pro jazyk Java, který využívá specifického formátování komentářů v kódu.

[43] Line graph je jeden z druhů vizualizací v Cognos Analytics.

[44] Myšleno od verze Cognos 11.

[45] Mezi ostatní datové zdroje mohou patřit např. dedikované kostky a soubory.



Obrázek 6.2: Ukázka vizualizace datových toků v IBM Cognos

- **Pokrytí dashboardů a stories** – prozatím jsou zpracovávány jen nejrozšířenější typy dokumentů, tj. reporty, případně interaktivní reporty. Mezi další typy dokumentů patří Dashboardy a Stories, jejichž specifikace už modul extraktoru umí stáhnout, ale prozatím se s nimi nepočítá pro analýzu datových toků.

- **Pokrytí reportů s přímo vloženým sql** – než byly v Cognos 11 sjednoceny datové zdroje do datových modulů, byla i možnost přímo v reportu sestavit *Query* objekty, které obsahovaly definici SQL dotazu. Tyto *Query* lze navíc referencovat v jiných objektech tohoto typu a příslušné reference mají jinou formu, než když se tyto objekty navzájem referencují a mají za zdroj model.

- **Ostatní druhy kostek** – kromě dimenzionálně modelovaného FM modelu, pro který je analýza toků hotova, podporuje Cognos ještě další druhy OLAP kostek, mezi které patří především Dynamic Cubes a stále značně rozšířené Power Cubes. Získat jejich specifikace se zdá být komplikovanější, neboť jsou to produkty dedikovaných nástrojů a Cognos SDK práci s nimi nezahrnuje přímo. Bude ale třeba nalézt způsob získání potřebných metadat a pokrytí těchto objektů analytické vrstvy.

- **Sloupcově orientované soubory jako zdroje dat** – jako fyzické zdroje dat mohou být použity vedle databází a Power Cubes i textové soubory, nejčastěji ve formátu CSV. V prototypu pro ně v současné době není implementovaná analýza toků ani extrakce.

- **Rozlišování typů toku** – na řadě míst kódu byla zjednodušeno zpracování komplikovaných výrazů, které obsahují odkazy na jiné datové objekty. Následkem toho se často ztratila informace o typu toku, tedy zda jde o přímý datový tok, nebo filtrování. V budoucnu bude potřeba toto rozlišit pomocí robustnějšího systému parsování výrazů. Jako vhodným kandidátem pro pomoc s tímto problémem se jeví knihovna ANTLR, která je v obdobných případech v jiných modulech Manta Flow už využita.

- **Detailnější resolving vizualizací** – dle aktuální implementace jsou odhaleny datové toky na úrovni grafických prvků na stránce, tj. různých grafů, tabulek, apod. Už se však neřeší, jaké potomkovské objekty by se v nich ještě daly identifikovat a dodat tak toku větší detail. Implementace byla v tomto ohledu zjednodušena, protože objekty uvnitř různých druhů vizualizací mají značně heterogenní povahu a proto si pokrytí všech těchto typů vyžádá více času.

Connector Cognos

Module	Fail
eu.profinit.manta:manta-connector-cognos	0
eu.profinit.manta:manta-connector-cognos-extractor	0
eu.profinit.manta:manta-dataflow-generator-cognos	0

Obrázek 6.3: Znamení, že testy na Jenkinsu prochází.

6.6 Testování

Testování prototypu je z větší části založeno na spouštění složitějších scénářů a komplexních celků, proto bych je označil převážně za integrační minimálně v kontextu integrace jednotlivých balíčků. Testy jsou založeny na kontrolách výsledků a výjimek, které scénáře produkují. Tento přístup je na několika místech doplněn testy omezených částí, které ale většinou nelze prohlásit za tradiční jednotkové testy. Jako zdrojová data slouží ukázkové sady reportů, modelů atd., které jsou pro IBM Cognos veřejně k dispozici. Testy jsou implementovány s použitím frameworku JUnit.

V této kapitole nelze nalézt žádné testy *connector-model*, což je inherentně způsobenou povahou tohoto balíku. Obsahuje totiž pouze rozhraní, nikoliv jejich implementaci. Testy pro zbytek modulů jsou rozebrány v následujících sekcích.

Všechny uvedené testy při současné implementaci úspěšně prochází, jak ilustruje obrázek 6.3 z nástroje Jenkins[46], který Manta pro nasazování nových verzí používá.

6.6.1 Testy Extractoru

Testování extraktoru kromě implementovaných unit testů v mnoha případech spouští i samotnou extrakci. Stahovány mohou být specifikace ukázkových modelů o desítkách MB, což může prodloužit dobu průběhu testů do řádu minut.

V současné implementaci je většina testů v extraktoru ignorována, protože ještě není k dispozici instalace Cognos Analytics na vývojářském serveru. Navíc se při vývoji hodí nespouštět několikaminutové testy. Lokálně ale při spuštění všechny prochází.

ConnectionTest pokrývá kontrolu připojení ke Content Store, a to anonymním i konkrétním uživatelem. Ověřováno je vyhození příslušných

[46] Jenkins se používá pro nasazování aplikace a průběžnou integraci (continuous integration).

výjimek při zadání špatné URL[47] adresy dispatcheru i špatné heslo nebo neexistující uživatel.

ExtractionTest se stará o ověření, že proběhla extrakce požadovaných souborů. V testech jsou zkoušeny různé kombinace extrahovaných objektů. Test je přeskočen při neúspěšném připojení ke Content Store, protože tuto funkčnost již pokrývá *ConnectionTest*.

SpecificationTest kontroluje, že se specifikace objektů uložily korektně a s přidanými informacemi. Nechybí také testovací metoda, ve které je pro všechny reporty založené na modelech verifikováno, že je příslušný model extrahován.

ContentStoreUtilsTest ověřuje korektní chování utility[48] třídy *ContentStoreUtil*, která skrze několik statických metod pomáhá pracovat s Content Store a Cognos Searchpath. V testech se primárně kontroluje správný překlad řetězců a výjimky požadovaného typu.

6.6.2 Testy Konektoru

Balík *connector* obsahuje pouze dvě třídy – *CogInputReader* a *ProviderCache*, jejichž primárním úkolem je načtení extrahovaných souborů do struktury vhodné pro generování grafu datových toků. Jako vstupní data pro testy obsažené v tomto balíku slouží ověřené extrahované specifikace.

LoadTest se stará o kontrolu průběhu načítání extrahovaných reportů a všech ostatních souborů, na kterých závisí vytvoření funkční jednotky *DataFlowInput*, určené jako vstup balíku pro generování datových toků. Je ověřeno, že všechny závislé soubory se vyskytují v extrahovaných složkách na požadovaných místech.

ProviderCacheTest je především test pro třídu *ProviderCache*. Ověřuje se, že cachování funguje očekávaným způsobem, neprobíhá redundantní resolving objektů a chování při překročení limitu uložených objektů.

6.6.3 Testy Resolveru

Testy modulu resolveru jsou trochu nelogicky umístěny u testů konektoru. Tato nepříjemnost je důsledkem několika faktorů. Hlavní příčinou je potřeba vyhnout se cyklické závislosti balíčků *connector* a *connector-resolver*. Konektor je na resolveru nutně závislý a pokud bych chtěl využít během testování resolveru načítací funkce konektoru, vznikl by zmíněný cyklus. Alternativním

[47] URL (Uniform resource locator) je adresa nějakého zdroje na webu včetně použitého protokolu.

[48] Utility class je označení pro třídu, která obvykle má pouze statické metody a proměnné a slouží jako pomocná třída pro několik jiných tříd.

řešením tohoto problému by bylo částečně zduplikovat funkce konektoru v testech resolveru. Zvolené umístění testů resolveru v balíčku konektoru mi přijde jako menší provinění proti principům softwarového inženýrství. Tím spíš, pokud většina testování je pojata jako testování komplexních celků spíše než jednotek.

Následující výčet tedy reálně testuje funkčnost *connector-resolver*, ale v kódu je k nalezení v adresáři s testy balíku *connector*.

ReportStructureTest testuje, zda se uložená specifikace korektně transformovala na objekt *Report*. Z vybraných specifikací reportů byly manuálně získány statistiky počtu náhodně vybraných potomků a souvisejících objektů. Vůči těmto statistikám se automaticky kontroluje výsledná struktura získaného reportu.

ModelStructureTest je obdobný jako *ReportStructureTest* s tím rozdílem, že pracuje s FM modely.

6.6.4 Testy Dataflow generatoru

Testy tohoto modulu předpokládají korektní funkčost třídy *Graph*, která je ověřena v příslušném Manta balíku. Testy v balíku *dataflow-generator* ověřují především správné generování grafu podle vybraných objektů a jejich vnitřních referencí. Test napojení FM modelů na externí datové zdroje zatím není implementován, ale do budoucna se s ním rozhodně počítá.

ModelNodesTest ověřuje na náhodně vybraných vnitřních objektech FM modelů, že uzly, které mají dané objekty reprezentovat, byly v grafu vytvořeny. Kontroluje, zda byly také vytvořeny hrany mezi vzájemně se referencujími *Query Subject Itemy* a také mezi *Calculation* a jejími zdrojovými *Query Subject Itemy*.

ReportNodesTest funguje na podobném principu jako *ModelNodesTest*. Na náhodně vybraných objektech v reportech ověřuje, že vznikly příslušné uzly a hrany. Explicitně testuje existenci hran mezi jednotlivými *Report Itemy* a *Query Itemy*.

Závěr

Tato diplomová práce si kladla dva hlavní cíle. Prvním z nich bylo na základě analýzy reportingových nástrojů Cognos, Excel, SSRS a OBIEE identifikovat datové objekty v těchto nástrojích a vymyslet jednotný model typů uzlů pro graf datových toků.

Během procesu bylo zjištěno, že mít stoprocentně sjednocený model těchto uzlů není zcela možné ani žádoucí. Byla by totiž značně redukována autentičnost nástrojů, což by se mj. mohlo projevit zmatením uživatelů, kteří by často nacházeli významově podobné, ale přesto neznámé objekty. Také programátoři by se museli snažit uměle uzly mapovat, což by opět mohlo mít za následek ztrátu významu.

Místo plně sjednoceného modelu byla proto navržena sada uzlů, které typicky mají napříč reportovacími nástroji své analogie. Nástroje ale nebudou nuceny tyto uzly striktně využívat. Klíčové bude povědomí implementujícího technika o daných uzlech a schopnost většinu proprietárních typů uzlů na uzly v navrženém modelu namapovat pro účely integrace s programy třetích stran. Příklad mapování pro Cognos byl v dané kapitole uveden.

Druhým důležitým úkolem byla implementace prototypu konektoru pro nástroj Manta Flow, který měl analyzovat datové toky v IBM Cognos. Tento modul byl úspěšně navržen a implementován v rozsahu dostatečném pro brzké experimentální nasazení u zákazníků. Implementace tedy pokrývá celý proces typický pro konektor. Dokáže z reportingového nástroje získat potřebná data, transformovat je na interní model a vygenerovat z něj graf datových toků. Konektor byl také řádně otestován.

Zároveň byla navržena další rozšíření prototypu, která bude vhodné implementovat pro zvýšení pokrytí datových toků v tomto nástroji a podporu všech typů reportů a objektů analytické vrstvy.

Osobně jsem si z celého projektu odnesl mnoho cenných zkušeností s dlouhodobě vyvíjeným a uznávaným produktem a množstvím technologií. Věřím, že tato práce i implementovaný prototyp budou cennou inspirací pro další podobné konektory reportingových nástrojů.

Literatura

[1] Collins, J. C.: Microsoft Excel: Create a picture-based dashboard report. *Houston Chronicle [online]*, srpen 2018, [cit. 2019-3-31]. Dostupné z: https://www.journalofaccountancy.com/content/jofa-home/issues/2018/aug/excel-picture-based-dashboards/_jcr_content/contentSectionArticlePage/article/articleparsys/image.img.jpg/1532553816302.jpg

[2] IBM Cognos Mobile 10.2.2. *Element61 [online]*, [cit. 2019-3-31]. Dostupné z: https://www.element61.be/sites/default/files/cognosmobile-figure-6-active-reports-offer-an-interactive-appealing-way-of-presenting-data.png?width=500&height=500

[3] Manta Flow Pricing, Features, Reviews and Comparison of Alternatives. *GetApp [online]*, [cit. 2019-4-8]. Dostupné z: https://www.getapp.com/business-intelligence-analytics-software/a/manta-flow/#gallery-1

[4] Hermann, L.; Ulrych, J.; Hollinger, R.: Manta Flow Architecture. *Manta confluence (interní dokumentace)*, duben 2019, [cit. 2019-4-17]. Dostupné z: https://mantatools.atlassian.net/wiki/spaces/MTKB/pages/70230122/Manta+Flow+Architecture

[5] Zaigraev, A.: Multidimensional Database concepts: Measure. *javamagic.blog*, červenec 2018, [cit. 2019-4-26]. Dostupné z: https://javamagic.blog/2018/07/03/multidimensional-database-concepts-measure/

[6] Star schema. *Wikipedia*. [online], [cit. 2019-4-21]. Dostupné z: https://en.wikipedia.org/wiki/Star_schema

[7] Leonard, K.: The Role of Data in Business. *Houston Chronicle [online]*, říjen 2018, [cit. 2019-3-24]. Dostupné z: https://smallbusiness.chron.com/role-data-business-20405.html

[8] Hartman, L.: 3 Reasons Why Data Governance is Important. *Semarchy [online]*, říjen 2018, [cit. 2019-3-24]. Dostupné z: https://blog.semarchy.com/3-reasons-why-data-governance-is-important

[9] What is Business Intelligence? Definition and Example. *Guru99*. [online], [cit. 2019-3-30]. Dostupné z: https://www.guru99.com/business-intelligence-definition-example.html

[10] Richardson, M.: 7 Reasons Business Intelligence Is Vital To Business Success. *Maximizer [online]*, září 2018, [cit. 2019-3-31]. Dostupné z: https://www.maximizer.com/blog/7-reasons-why-business-intelligence-is-vital-to-business-success/

[11] Reporting Tools: Everything You Need to Know. *JReport [online]*, [cit. 2019-3-31]. Dostupné z: https://www.jinfonet.com/resources/bi-defined/reporting-tools/

[12] Serra, J.: Why use a SSAS cube? *James Serra's Blog*, srpen 2013, [cit. 2019-4-5]. Dostupné z: https://www.bitool.net/software/analysis-services.html

[13] Microsoft SQL Server Analysis Services. *Business Intelligence, Tools and software*. [online], [cit. 2019-4-5]. Dostupné z: https://www.bitool.net/software/analysis-services.html

[14] Sedlák, J.: V Dejvicích roste další velká česká softwarová věc. *Connect.cz*, únor 2015, [cit. 2019-4-5]. Dostupné z: https://connect.zive.cz/clanky/v-dejvicich-roste-dalsi-velka-ceska-softwarova-vec/sc-320-a-177282

[15] Cyprich, P.: Rozhodnuto. S Czech ICT Alliance jedou do Silicon Valley dva startupy - Manta a realPad. *tyinternety.cy*, únor 2015, [cit. 2019-4-5]. Dostupné z: https://tyinternety.cz/startupy/rozhodnuto-s-ict-alliance-jedou-silicon-valley-dva-startupy-manta-realpad/

[16] What is Dataflow? Definition from Techopedia. *Techopedia*. [online], [cit. 2019-4-7]. Dostupné z: https://www.techopedia.com/definition/6743/dataflow

[17] What is Data Lineage? *Octopai*, prosinec 2017, [cit. 2019-4-10]. Dostupné z: https://www.octopai.com/what-is-data-lineage/

[18] Directed graph. *Wikipedia*. [online], [cit. 2019-4-21]. Dostupné z: https://en.wikipedia.org/wiki/Directed_graph

[19] Sýkora, J.: *Incremental update of data lineage storage in a graph database*. Diplomová práce, České vysoké učení technické v Praze, Fakulta informačních technologií, 2018.

[20] Automate your data lineage. *Manta*. [online], [cit. 2019-4-10]. Dostupné z: https://getmanta.com/

[21] What Is Impact Analysis? *bizfluent*, říjen 2018, [cit. 2019-4-17]. Dostupné z: https://bizfluent.com/info-7809817-impact-analysis.html

[22] Automate your data lineage. *Manta*. [online], [cit. 2019-4-10]. Dostupné z: https://ocean.getmanta.com/index.php/s/DyRE6YDJmCiKXGQ#pdfviewer

[23] Root Cause Analysis. *Washington State Department of Enterprise Services*. [online], [cit. 2019-4-17]. Dostupné z: https://des.wa.gov/services/risk-management/about-risk-management/enterprise-risk-management/root-cause-analysis

[24] Automate your data lineage. *Manta*. [online], [cit. 2019-4-17]. Dostupné z: https://ocean.getmanta.com/index.php/s/0x2eK9QHik4ZadT#pdfviewer

[25] Automate your data lineage. *Manta*. [online], [cit. 2019-4-17]. Dostupné z: https://ocean.getmanta.com/index.php/s/eEVwLkIzOyVnVrH#pdfviewer

[26] OLAP KOSTKY V KOSTCE. *blog.helios.eu*, 2018, [cit. 2019-4-26]. Dostupné z: https://blog.helios.eu/cz/clanky/olap-kostky-v-kostce/

[27] Star and SnowFlake Schema in Data Warehousing. *Guru99*, [cit. 2019-4-26]. Dostupné z: https://www.guru99.com/star-snowflake-data-warehousing.html

[28] Kotrč, J.: SSRS. *Manta confluence (interní dokumentace)*, duben 2019, [cit. 2019-5-1]. Dostupné z: https://mantatools.atlassian.net/wiki/spaces/MT/pages/475693057/SSRS

[29] Watson, E.: Introduction to SSIS. *SQLSaturday*, [cit. 2019-5-2]. Dostupné z: https://www.sqlsaturday.com/SessionDownload.aspx?suid=13363

[30] Gogia, S.: What is SSRS? *Quora*, listopad 2016, [cit. 2019-5-1]. Dostupné z: https://www.quora.com/What-is-SSRS

[31] Kotrč, J.; Tornóci, M.: SSAS. *Manta confluence (interní dokumentace)*, duben 2019, [cit. 2019-5-1]. Dostupné z: https://mantatools.atlassian.net/wiki/spaces/MT/pages/279576581/SSAS

[32] Buldyk, Y.: OBIEE. *Manta confluence (interní dokumentace)*, duben 2019, [cit. 2019-5-3]. Dostupné z: https://mantatools.atlassian.net/wiki/spaces/MT/pages/610238465/OBIEE

[33] Kumar, V.: What is OBIEE? *Quora*, leden 2019, [cit. 2019-5-4]. Dostupné z: https://www.quora.com/What-is-OBIEE

[34] Míček, D.: *Analýza datových toků v Excelu*. Diplomová práce, České vysoké učení technické v Praze, Fakulta informačních technologií, 2019 (půjde k obhajobě).

[35] Introduction to Microsoft Excel. *Guru99*, [cit. 2019-4-27]. Dostupné z: https://www.guru99.com/introduction-to-microsoft-excel.html

[36] What Is an ODBC Driver? *Progress*, [cit. 2019-4-26]. Dostupné z: https://www.progress.com/faqs/datadirect-odbc-faqs/what-is-an-odbc-driver

[37] Bruns, D.: Understanding data series. *ExcelJet*, [cit. 2019-4-27]. Dostupné z: https://exceljet.net/lessons/understanding-data-series

[38] Cross-Tabulating Variables: How to Create a Contingency Table in Microsoft Excel. *TurboFuture*, srpen 2014, [cit. 2019-4-27]. Dostupné z: https://turbofuture.com/computers/Cross-Tabulating-Variables-How-to-Create-a-Contingency-Table-in-Microsoft-Excel

[39] Pivot Cache in Excel: What Is It and How to Best Use It. *Trump Excel*, 2015, [cit. 2019-4-27]. Dostupné z: https://trumpexcel.com/pivot-cache-excel/

[40] What is Cognos? Definition form Techopedia. *Techopedia*, [cit. 2019-5-1]. Dostupné z: https://www.techopedia.com/definition/14673/cognos

[41] Content Store. *IBM Knowledge Center*, [cit. 2019-5-1]. Dostupné z: https://www.ibm.com/support/knowledgecenter/en/SSEP7J_10.1.1/com.ibm.swg.ba.cognos.crn_arch.10.1.1.doc/c_arch_contentstore.html

[42] Currie, D.: Digging into IBM Cognos Dynamic Cubes. *Business Intelligence, Big Data and Cognos Analytics*, srpen 2017, [cit. 2019-5-1]. Dostupné z: https://www.davidpcurrie.com/digging-into-ibm-cognos-dynamic-cubes/

[43] TIOBE Index for April 2019. *TIOBE*. [online], [cit. 2019-4-20]. Dostupné z: https://www.tiobe.com/tiobe-index/

[44] What is Java Bytecode? *Javatpoint*. [online], [cit. 2019-4-20]. Dostupné z: https://www.javatpoint.com/java-bytecode

[45] Software development kit guide - introduction. *IBM Knowledge Center*. [online], [cit. 2019-4-20]. Dostupné z: https://www.ibm.com/support/knowledgecenter/en/SSEP7J_11.0.0/com.ibm.swg.ba.cognos.ca_dg_sdk.doc/c_introduction_sdk.html#id_Introduction_sdk

[46] Lowe, D.: What you need to know about JavaBeans. *dummies.com*, [cit. 2019-4-20]. Dostupné z: `https://www.dummies.com/programming/java/need-know-javabeans/`

Příloha **A**

Seznam použitých zkratek

API Application Programming Interface

ANTLR Another Tool for Language Recognition

BI Business Intelligence

CLI Command Line Interface

CSV Comma Separated Values

DAX Data Analysis Expressions

DGC Data Government Center

FM Framework Manager

GDPR General Data Protection Regulation

IBM International Business Machines

IGC Information Governance Catalog

JSON Javascript Object Notation

MB Megabyte

MDX Multi Dimensional Expressionos

MSSQL Microsoft Structured Query Language

OBIEE Oracle Business Intelligence Enterprise Edition

OLAP Online Analytical Processing

REST Representational State Transfer

RTF Rich Text Format

SDK Software Development Kit

SOAP Simple Object Access Protocol

SQL Structured Query Language

SSAS SQL Server Analysis Services

SSIS SQL Server Integration Services

SSRS SQL Server Reporting Services

TIOBE The Importance of Being Earnest

UML Unified Modeling Language

XLS Excel Spreadsheet

XML Extensible Markup Language

Příloha **B**

Obsah přiloženého CD

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