ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Nástroj pro extrakci metadat z databáze DB2
Student:	Miroslav Špak
Vedoucí:	Ing. Michal Valenta, Ph.D.
Studijní program:	Informatika
Studijní obor:	Softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2017/18

Pokyny pro vypracování

Pro statickou analýzu SQL skriptů je potřeba znát informace o objektech existujících v relační databázi, s níž daný skript pracuje. Struktura objektů, rozsah metadat a způsob jejich získání se liší podle použité databáze.
1. Popište strukturu metadat v databázi DB2 a srovnejte ji s metadaty v Oracle a MS SQL Server.
2. Vypracujte seznam metadat, která je pro statickou analýzu potřeba z této databáze extrahovat, ověřte jejich dostupnost a identifikujte možné způsoby, jak tato metadata z databáze extrahovat.
3. Na základě analýzy navrhněte nástroj pro extrakci metadat z databáze DB2.
4. Implementujte prototypové řešení extrahující metadata do existující datové struktury projektu Manta. Použijte vhodnou architekturu umožňující zapojení nástroje do statické analýzy SQL skriptů. Pro přístup k databázi použijte vhodný framework.
5. Vytvořte sadu testovacích dat a automatické testy ověřující správnou funkčnost implementovaného řešení. Vytvořte uživatelskou dokumentaci nástroje.

Seznam odborné literatury

Dodá vedoucí práce.

L.S.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

prof. Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 23. listopadu 2016

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra softwarového inženýrství

Bakalářská práce

Nástroj pro extrakci metadat z databáze DB2

Miroslav Špak

Vedoucí práce: Ing. Michal Valenta, Ph.D.

9. května 2017

Poděkování

Rád bych poděkoval vedoucímu této práce Ing. Michalu Valentovi Ph.D. za jeho pomoc a hodnotné rady. Také bych rád poděkoval všem členům z projektu Manta, jmenovitě hlavně Mgr. Jiřímu Touškovi, nejen za pomoc s pochopením jejich softwaru.

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., autorského zákona, ve znění pozdějších předpisů.

V souladu s ust. § 2373 občanského zákoníku tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla a za jakýmkoli účelem (včetně užití k výdělečným účelům), vč. možnosti Dílo 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 9. května 2017 .....

Odkaz na tuto práci

Špak, Miroslav. *Nástroj pro extrakci metadat z databáze DB2*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2017.

Abstrakt

Práce se zabývá analýzou metadat databáze IBM DB2, návrhem nástroje pro jejich extrakci a jeho implementací. Nejdříve se věnuje popisu struktury metadat a jejímu srovnání se dvěma zástupci konkurenčních databází. Dále řeší dostupnost a způsoby, jak tato metadata z databáze extrahovat. Na základě této analýzy následuje návrh a implementace prototypového řešení nástroje a testy ověřující jeho správnou funkčnost.

Klíčová slova extrakce metadat, databáze, IBM DB2, Manta

Abstract

The thesis is dealing with metadata analysis in IBM DB2 database, design and implementation of the tool for its extraction. At first it is focused on describing the structure of metadata and compares it with two representatives of competitive databases. Also methods of extraction and metadata accessibility are discussed. Based on this analysis the tool prototype is designed and implemented. At last its flawless function is tested.

Keywords metadata extraction, database, IBM DB2, Manta

Obsah

Seznam obrázků

Úvod

Moderní informační systémy pracují s ohromným množstvím informací a téměř každý potřebuje svá data ukládat. To je důvod, proč se databáze staly běžnou součástí dnešních systémů. Stejně tak, jako existuje mnoho různých aplikací, je i spousta různých databází. Z toho vzniká problém, jak z nich jednotně získávat informace o uložených datech. Takovými informacemi může být způsob, jakým jedna data ovlivňují jiná, odkud kam se přelévají, na čem závisí atd. Tyto informace o datech, takzvaná metadata, mohou být zajímavá zejména pro velké firmy, například pro lepší správu firemních informací nebo jejich zabezpečení.

Tímto problémem se zabývá projekt Manta, kterému tato práce vytvoří prototypové řešení nástroje pro extrakci metadat z databáze IBM DB2. Manta dnes umí pracovat s různými databázemi, ale podporu pro DB2 prozatím nemá, což je důvodem k volbě tohoto tématu.

Cíl práce

Práce se zabývá analýzou metadat databáze IBM DB2, návrhem nástroje pro jejich extrakci a jeho implementací.

Nejprve se zaměří na popis struktury metadat v DB2 a srovná ji s metadaty v databázích Oracle a MS SQL Server, poté diskutuje minimální práva, která jsou nutná pro přístup k těmto metadatům. Přitom vznikne seznam metadat nutných pro statickou analýzu nástrojem Manta doplněný o možné způsoby jejich extrakce.

Po výše zmíněné analýze přejde k návrhu nástroje, kde zvolí nejlepší z postupů pro extrakci jednotlivých metadat. Následuje imlementace prototypového řešení nástroje s otestováním jeho správné funkčnosti.

Kapitola 1

Vymezení problematiky

Přistupovat k metadatům a číst je z databáze dnes potřebuje nejeden nástroj. Jedním příkladem za všechny může být jakýkoli databázový klient. Takovýto databázový klient potřebuje získat metadata z databáze minimálně pro správné zobrazení názvů schémat, tabulek, sloupců nebo pro náhledy uložených procedur, jejich případné ladění a jiné.

Klientů, kteří přistupují k metadatům databází, je nepřeberné množství. To je možné díky tomu, že přístup k metadatům bývá dobře dokumentován samotným výrobcem databáze. Na jeho stránkách lze najít doporučené postupy, mnohé rady a řešení různých problémů.

Práce se bude zabývat jak těmito doporučenými postupy, tak i některými výrobcem nepodporovanými způsoby, které si našly oblibu u programátorů a mohou být z některého hlediska výhodné.

1.1 Data lineage

Extrakce metadat v této práci se používá ke sledování tzv. „datové linie“ (data lineage). „*Pojem data lineage v Encyclopedia of Database Systems odkazuje k pojmu ‚původ dat‘ (Data Provenance).*“[2, str. 187]
„*Termín ‚Data Provenance‘ odkazuje na záznam kroků, které objasňují původ části dat (v databázi, dokumentu nebo repozitáři) společně s vysvětlením, jak a proč se data dostala do současného stavu.*“[3, překlad vlastní]

1.2 Manta

Vizualizaci těchto datových linií provádějí právě nástroje Manta (viz obrázek 1.1). Z pohledu Manty je nutné mít vlastní implementaci nástroje na extrakci metadat pro každou z podporovaných databází. Tato potřeba vychází z velké rozmanitosti všech databázových systémů a z mnoha rozdílů ve způsobech

ukládání nebo přístupu k metadatům. Komplexnost jednoho nástroje, který by obsáhl každou podporovanou databázi, by byla neudržitelná.

Obrázek 1.1: Vizualizace pomocí Manta Flow[1]

1.3 Databáze DB2 a Manta

V současné době neexistuje nástroj na extrakci metadat z databáze DB2 pro projekt Manta. Lze proto vycházet pouze z již implementovaných řešení pro jiné databáze.

Kapitola 2

Analýza struktury metadat

Typy objektů, které je možné ukládat v různých databázích, se mohou lišit. Rozdílná může být navíc jejich hierarchie nebo struktura. To je také jeden z důvodů, proč se v různých implementacích databází liší i struktura jejich metadat. Odlišné mohou být způsoby získání metadat, kdy jedna databáze nabízí strukturované informace o každém uloženém typu objektu a jiná u některých pouze SQL skripty DDL (Data Definition Language) příkazů použitých k jejich vytvoření (dále souhrně jen „DDL“). Typicky jsou informace o objektech databáze uložené v souboru tabulek nebo databázových pohledů (views) nazývaném systémový katalog. Výjimečně se setkáme s jiným způsobem přístupu k těmto informacím.

Analýza se zabývá strukturou metadat v databázích Oracle, Microsoft SQL Server a IBM DB2.

2.1 Oracle

Databáze Oracle nabízí kromě typických tabulek a pohledů několik typů uživatelem definovaných objektů. Jsou to například balíčky (packages), proměnné (variables), kurzory (cursors), pole (arrays), objektové typy (OT) a jiné.

Většina objektů v databázi je uložena ve schématech. Každé schéma nese název uživatele, který jej vytvořil. Proto se v katalogu názvy schémat získají z uživatelských jmen.

Informace o objektech v databázi je možné najít v systémovém katalogu. Systémový katalog je soubor pohledů uložených v systémovém schématu s názvem SYS. Všechny pohledy v tomto schématu vlastní systémový uživatel SYS a jsou určeny pouze pro čtení. Data uložená v katalogu se mění při zavolání DDL příkazu.

2.1.1 Rozdělení katalogu

Systémový katalog má mnoho pohledů rozdělených do tří kategorií podle prefixu jména:

- „*pohled USER – pro uživatele (co je v uživatelově schématu),*
- *pohled ALL – jako rozšíření pohledu pro uživatele (vše, k čemu může uživatel přistupovat),*
- *pohled DBA – pro databázového administrátora (objekty všech schémat v databázi).*“[4, překlad vlastní]

Každá kategorie nabízí různý pohled na stejná data. Nejvhodnější kategorií pro extrakci metadat jsou pohledy DBA. „*Pohledy s prefixem DBA poskytují globální náhled na celou databázi.*“[4, překlad vlastní]

Pohledy katalogu jsou dále děleny podle objektů, o kterých drží informaci. Například v *_TABLES (* značí jeden z prefixů) lze najít informace o tabulkách, v *_OBJECTS uživatelsky definované objekty a závislosti mezi procedurami, funkcemi a dalšími v *_DEPENDENCIES atd. Pro tyto pohledy platí, že mají takové sloupce, které popisují všechna důležitá metadata objektů.

2.2 Microsoft SQL Server

SQL Server nabízí několik různých způsobů přístupu k metadatům. Stejně jako je tomu u Oracle, každá databáze SQL Serveru nabízí katalogové schéma. Navíc také nabízí schéma s názvem INFORMATION_SCHEMA. „*Pohledy informačního schématu jsou jednou z několika metod, které SQL Server nabízí pro přístup k metadatům. … Tyto pohledy obsažené v SQL Serveru jsou v souladu s definicí standardu ISO pro informační schéma.*“[5, překlad vlastní]

Metody přístupu k metadatům:

- pohledy informačního schématu,
- systémový katalog,
- systémové procedury a funkce pro práci s metadaty.

2.2.1 Informační schéma

Pohledy informačního schématu popisují hlavní objekty databáze, jako jsou tabulky, pohledy, schémata, datové typy (zde domains) a procedury. Bohužel některé z objektů nejsou v těchto pohledech k nalezení, například spouštěče (triggery). Pro přístup k těmto objektům je nutné použít jinou z již zmiňovaných metod.

2.2.2 Systémový katalog

Microsoft doporučuje k získání metadat použití katalogových pohledů. Tyto pohledy se nacházejí v systémovém schématu s názvem SYS. Pohledy systémového katalogu jsou děleny podle typů objektů stejně, jako je tomu u databáze Oracle. „*Některé katalogové pohledy dědí řádky z jiných katalogových pohledů. Například katalogový pohled SYS.TABLES dědí ze SYS.OBJECTS.*“[6, překlad vlastní]

2.3 IBM DB2

V databázi DB2 je také možnost definování uživatelských typů objektů velice podobných, jako je tomu u databáze Oracle. Všechny objekty v databázi, s výjimkou objektů typu server (odkaz do jiné databáze), se ukládají do schémat. Ta je možné rozdělit do dvou kategorií: systémová a uživatelská. Názvy systémových schémat začínají prefixem SYS, ten je rezervovaný a nelze ho použít pro uživatelská. Jedním ze systémových schémat je SYSCAT, které plní úlohu systémového katalogu a data objektů přístupná pomocí tohoto schématu jsou právě ta popisující databázi.

Databáze navíc dovoluje sdružování objektů do modulů a také vytváření takzvaných typovaných tabulek nebo pohledů, které mají strukturu některého z uživatelem definovaných objektů řádkového typu (dále pouze jako typ row). Objekty mohou mít ve většině případů také svoje aliasy. To vše značně komplikuje celou strukturu metadat v databázi.

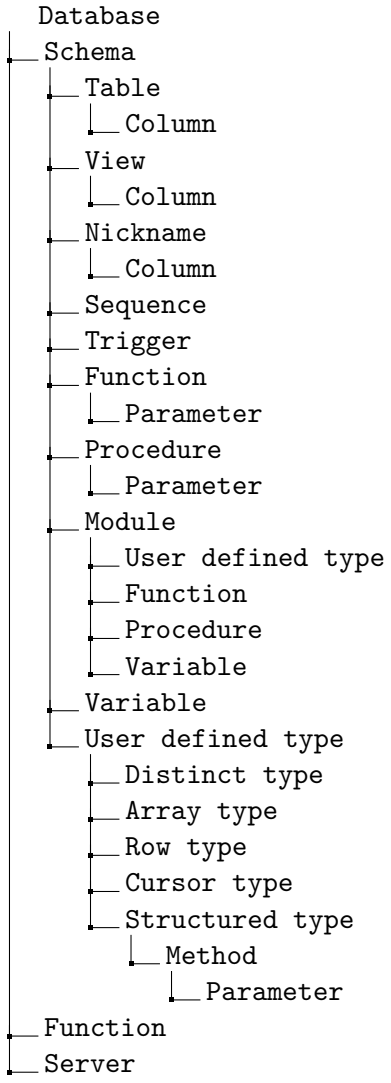
2.3.1 Kompatibilita s databází Oracle

Specialitou databáze DB2 je kompatibilita s databází Oracle. „*DB2 poskytuje funkce umožňující aplikacím, které byly napsány pro databázi Oracle, být použity v databázi DB2 bez nutnosti jejich přepsání.*“[7, překlad vlastní] Při aktivaci kompatibility lze s pár omezeními používat i příkazy pro tvorbu objektů, které používá databáze Oracle.

Objekty vytvořené v tomto režimu jsou namapovány na objekty databáze DB2 se stejnými nebo alespoň podobnými vlastnostmi. Oracle balíčky jsou mapovány na moduly, záznamy (objekty typu record) na typ row a Oracle kolekce na pole. Problém působí zejména kurzory. Databáze Oracle má dva typy kurzorů – ref cursor a cursor, zatímco DB2 používá pouze jeden typ uživatelsky definovaného kurzoru. Proto Oracle ref cursor je reprezentován jako kurzor, ale pro Oracle cursor je vytvořena systémová funkce, která ukládá hodnoty do stejnojmenné proměnné.

2.3.2 Typy objektů

V rámci jedné instalace DB2 jsou vytvářeny databáze, ty jsou vrcholem hierarchie objektů. V každé takové databázi jsou objekty ukládány do schémat. Takovým objektem může být tabulka, pohled, uživatelem definovaný typ, modul aj. (viz obrázek 2.1).



Obrázek 2.1: Hierarchie objektů v DB2

2.3.3 Struktura metadat DB2

Schéma systémového katalogu obsahuje pohledy logicky rozdělené podle typů objektů. Každý pohled poskytuje informace o objektech všech schémat, na-

příklad TABLES, ve kterém je možné najít informace o všech existujících tabulkách (včetně systémových). Všechny uživatelsky definované typy mají jediný pohled se spoustou sloupců. U každého záznamu je jich mnoho s prázdnou hodnotou, protože informace poskytovaná sloupcem není k danému typu relevantní. O jakém typu objektu má každý záznam informaci, lze zjistit ze sloupce METATYPE. U složitějších typů, jako je například objektový typ, jsou v tomto pohledu pouze základní data. Informace o jeho položkách, metodách a o hierarchii jsou dostupné v jiných pohledech.

Metadata objektů vytvořených v režimu kompatibility jsou většinou totožná s metadaty objektů, na které se mapují. Výjimku tvoří moduly vytvořené z Oracle balíčků. Procedury a funkce těchto modulů nemají v katalogu vyplněné informace o jejich DDL. Další výjimkou jsou objekty typu Oracle cursor.

2.4 Shrnutí

Použité principy jsou ve všech srovnávaných databázích podobné. Všechny přistupují k poskytování metadat podobným způsobem a to pomocí pohledů systémového katalogu, ve kterých je možné nalézt téměř všechna potřebná metadata. Bohužel zde není žádný jednotný způsob pojmenování ani členění takového katalogu. Databáze se liší v přístupu k DDL skriptům. Zatímco v databázích Oracle a SQL Server se pro jejich získání použijí systémové procedury, DB2 poskytuje některá DDL ve svých katalogových pohledech.

Kapitola 3

Extrakce metadat

Pro statickou analýzu SQL skriptů je potřeba znát metadata objektů zmíněných dříve (viz obrázek 2.1). Dále je nutné extrahovat samotné SQL skripty.

Získání metadat objektů je možné uskutečnit dvěma způsoby. První ze způsobů spočívá v extrakci DDL skriptů použitých k vytvoření databáze a všech jejích objektů a v následné syntaktické analýze těchto skriptů právě za účelem zjištění typů objektů a jejich struktury (tzv. parsování). Tento způsob není příliš vhodný, pokud existuje jiné řešení. Je nutné několikrát projít všechny skripty a při každém průchodu si označit objekty s dostatkem známých informací.

Druhým způsobem je získání informací z databázového katalogu. DB2 nabízí pestrý katalog, a proto zde bude jako primární volba. Pokud není řečeno jinak, všechny pohledy se nacházejí ve schématu katalogu s názvem SYSCAT.

3.1 Objekty databáze

3.1.1 Databáze a schémata

Informace o databázích vytvořených v rámci jedné instalace se v systémovém katalogu nenacházejí. Jedna databáze je kořenem hierarchie objektů v DB2.

Schémata mají svá metadata uložena v pohledu s názvem SCHEMATA. Lze zde najít názvy všech schémat v databázi. Krátké názvy schémat se doplní mezerami zprava. Tyto mezery v názvu nejsou podstatné.

3.1.2 Tabulky a pohledy

Metadata pro tabulky lze nalézt v pohledu TABLES. Jsou zde informace nejen o všech tabulkách, ale také o všech pohledech, dočasných tabulkách, jejich aliasech, přezdívkách (nicknames, nejedná se o aliasy) atd. Typy záznamů (zda se jedná o tabulku, pohled, ...) se rozlišují pomocí příznaku TYPE. Pohledy mají navíc také vlastní katalog s názvem VIEWS, ve kterém se nachází pouze

informace podstatné pro pohledy. Krátké názvy tabulek se opět doplní mezerami, stejně jako je tomu u schémat. Bohužel v ojedinělých případech jsou tyto mezery podstatné.

Katalog také nabízí pohledy TABDEP a VIEWDEP. V nich lze najít informace o závislostech mezi tabulkami a pohledy. V případě pohledů jsou zde informace, pro jakou tabulku jsou vytvořeny. Opět se nacházejí informace o pohledech jak v TABDEP, tak ve VIEWDEP.

Tabulky a pohledy v DB2 mohou mít svoji hierarchii. Hierarchické závislosti mezi tabulkami a pohledy je možné nalézt v pohledu HIERARCHIES. Jak je již zvykem, zda se jedná o tabulku či pohled je odlišeno pomocí sloupce METATYPE.

Metadata sloupců tabulek, pohledů a přezdívek jsou k nalezení v pohledu COLUMNS, který poskytuje všechny informace o datových typech sloupců a další.

3.1.3 Sekvence a spouštěče

Pohledy sekvencí SEQUENCES a spouštěčů TRIGGERS obsahují několik málo dat o těchto objektech. Pro sekvence jsou to například typ sekvence, jestli se jedná o sekvenci nebo její alias, jaká je počáteční hodnota sekvence a její krok. V případě spouštěčů se jedná o informace, kdy se mají spustit. Stejně jako v předchozích případech najdeme v katalogu pohled TRIGDEP se závislostmi spouštěčů na ostatních objektech.

3.1.4 Servery a přezdívky

Servery (odkazy do jiných databází) se nenacházejí v žádném ze schémat. Pohled SERVERS poskytuje informace jako jsou název, typ, či verze vzdálené databáze.

Každou vytvořenou přezdívku (nickname) je možné nalézt v TABLES, viz výše. Také mají vlastní pohled NICKNAMES s několika doplňujícími informacemi, například objekt typu server, pro který jsou definovány, nebo název vzdálené tabulky.

3.1.5 Moduly a Oracle balíčky

Všechny moduly, včetně Oracle balíčků, lze nalézt v MODULES. Kromě jména modulu je zde i jeho typ. Podle sloupce MODULETYPE je možné poznat, jestli se jedná o klasický DB2 modul, alias nebo Oracle balíček.

3.1.6 Procedury, funkce a metody

Pro signatury procedur, funkcí a metod (souhrnně routines) je v katalogu více pohledů. Procedury mají pohled PROCEDURES a metody a funkce FUNCTIONS. Také existuje společný pohled ROUTINES. V případě ROUTINES slou-

pec ROUTINETYPE rozlišuje, o jaký typ se jedná. Ve kterémkoli z těchto pohledů lze zjistit názvy routines a zda jsou externí nebo interní. U interních poskytují informaci o použitém dialektu, u externích jazyk, ve kterém jsou napsány.

DB2 podporuje přetěžování, proto je možné mít více routine se stejným názvem. Pro jejich odlišení se ukládá sloupec SPECIFICNAME, který musí být unikátní. Parametry je možné nalézt analogicky v pohledech s názvy FUNCPARMS, PROCPARMS nebo ROUTINEPARMS. Vždy je zde uvedeno SPECIFICNAME routine, ke které parametr patří, jeho pořadové číslo a datový typ.

3.1.7 Uživatelem definované typy a proměnné

Pro všechny uživatelsky definované typy existuje jediný pohled DATATYPES, který informuje o všech datových typech v databázi. To zahrnuje systémové a primitivní datové typy, jako jsou INTEGER, VARCHAR, ... Datové typy jsou rozděleny podle sloupce METATYPE. Ten říká, o jaký typ se jedná (například A – pole, C – kurzor, ...). K tomu patří také pohled DATATYPEDEP, ve kterém lze nalézt informace o závislostech datových typů mezi sebou, či na jiném objektu v databázi.

K strukturovanému objektovému typu patří také pohled ATTRIBUTES, odkud je možné získat strukturu jeho položek. Stejnou úlohu má pro typ row pohled s názvem ROWFIELDS.

3.2 DDL

Skripty DDL je možné z databáze extrahovat několika různými způsoby. Pro jejich získání je možné použít nástroj *db2look* vyvinutý výrobcem databáze. Další možnosti jsou pomocí katalogových pohledů nebo použití nedokumentované systémové procedury. V případě, že není možné získat DDL žádnou ze jmenovaných metod, lze za pomoci extrahovaných metadat objektu takový skript vytvořit.

3.2.1 Db2look

Nástroj *db2look* se spouští v příkazové řádce operačního systému, na kterém je databáze nainstalována. Pro spuštění používá práva přihlášeného uživatele (viz Databázová práva přístupu k metadatům 3.3). To z něj nečiní vhodnou volbu pro extrakci DDL, pokud existuje jiná možnost.

Vnitřně tento nástroj používá výše zmíněnou systémovou proceduru, tuto skutečnost potvrzují chybové hlášky, které se objevili při několika pádech nástroje.

3.2.2 Systémový katalog

Systémový katalog je vhodnou volbou pro extrakci DDL skriptů. Ve většině případů, pokud katalog nabízí DDL, nacházejí se ve sloupcích s názvem TEXT nebo BODY. Bohužel katalog nenabízí DDL pro všechny objekty v databázi, pouze pro spouštěče, pohledy a téměř všechny routines. Routines nemají v katalogu svá DDL v případě, že byly definovány uvnitř Oracle balíčku. Ta jsou k nalezení v systémové tabulce SYSMODULES schématu SYSIBM uvnitř sloupců s definicí balíčku, tj. SOURCEHEADER a SOURCEBODY.

Pro ostatní skripty je nutné použít jiný způsob extrakce.

3.2.3 Systémová procedura

Systémová procedura nemá žádnou dokumentaci. Narozdíl od Microsoft SQL Server a Oracle, které mají systémové procedury pro generování DDL, IBM ji nijak oficiálně nepodporuje. Zmínka o ní se objevila na fóru opensource databázového klienta *Dbeaver*, který ji vnitřně používá[8].

Dle vlastního testování používá ve většině případů procedura stejné vstupní parametry jako *db2look*. Je zde ale několik omezení:

- Proceduru musí v databázi alespoň jedou spustit uživatel s administrátorskými právy. Při prvním spuštění se vytvoří nové schéma SYSTOOLS s tabulkami DB2LOOK_INFO.
- Pokud procedura skončí svoji práci s chybou, v některých případech chybí srozumitelný popis toho, proč chyba nastala.
- Některé vstupní parametry, které fungují u *db2look*, způsobí chybu při běhu procedury.
- Nelze generovat DDL pro objekty typu server a přezdívka (nickname).

3.3 Databázová práva přístupu k metadatům

Každý uživatel v databázi DB2 musí být existujícím uživatelem operačního systému. Všechny nástroje pro správu databáze (to zahrnuje i *db2look*) se při spuštění přihlásí do databáze aktuálně přihlášeným uživatelem. Toto chování lze obejít systémovými příkazy pro spuštění nástrojů pod jiným uživatelem.

3.3.1 Katalogové pohledy

Pokud není databáze vytvořena s pomocí klíčového slova „restrict“, jsou systémovému katalogu přidělena veřejná práva a může ho číst každý uživatel. V opačném případě nejsou v databázi při vytvoření přidělena žádná práva.

Na rozdíl od databáze Oracle, kde pro čtení systémového katalogu existuje uživatelská role, DB2 žádnou takovouto roli nemá. Namísto rolí používá databázové autority, například DBADM – databázový administrátor. Autorita s nejnižšími právy, která může číst systémový katalog, je ACCESSCTRL – správce přístupů. Bohužel použití této autority není vhodné. Uživatel s touto autoritou může měnit přidělená práva na některé objekty v databázi ostatním uživatelům.

Je tedy nutné udělit uživateli právo pro čtení (SELECT) ze všech katalogových pohledů, která použije.

3.3.2 Db2look a systémová procedura

Pro použití nástroje *db2look* a systémové procedury je nutné přidělit práva pro čtení katalogovým a některým systémovým pohledům. Pro samotnou systémovou a několik pomocných procedur jsou nutná také práva pro jejich spuštění (EXECUTE).

Dále se již práce nezabývá databázovými právy. Úplný výčet práv viz příloha B.3.

3.4 Shrnutí

Informace o závislostech mezi objekty v databázi jsou roztříštěné ve spoustě katalogových pohledů. Existují případy, kdy se v katalogu databáze objevují chyby.

Možné zaznamenané chyby v katalogu:

- alias pro přezdívku (nickname) se přidá jako alias sekvence,
- alias modulu se objeví mezi tabulkami,
- katalog odkazuje na již neexistující typ.

Informace o metadatech jsou úhledně řazeny do stejnojmenných pohledů v katalogu. Více objektů, ať už ve stejném schématu, jiných schématech nebo modulech a také různých typů, může mít stejné jméno. Je tedy potřeba pečlivě vybírat informace o objektech, aby nedošlo k chybnému spojení jména a nesprávného objektu.

Kapitola 4

Návrh

Extraktor je navržen primárně pro práci s ostatními nástroji projektu Manta. Nicméně návrh zohledňuje možnost implementace jiné datové struktury (slovníku), či změny datové vrstvy aplikace, například čtení metadat ze souborů namísto databáze.

4.1 Požadavky

Extrakce metadat

Nástroj se připojí k databázi IBM DB2 LUW, ze které extrahuje všechna metadata objektů spolu se skripty použitými k jejich vytvoření (viz objekty zmíněné v kapitole Extrakce Metadat 3).

Naplnění slovníku

V průběhu extrakce se získanými metadaty naplní datová struktura nástroje Manta (metadatový slovník).

Výběr extrahovaných objektů

Je možné nastavit, které typy objektů se mají z databáze extrahovat. Objekty, jejichž metadata nejsou extrahována z databáze, nejsou přidány do slovníku.

Uložení extrahovaných skriptů

Extrahovaná DDL se uloží na disk do adresáře nastaveného před spuštěním samotné extrakce. V případě, že adresář neexistuje, vytvoří se. V opačném případě se nejprve vymaže jeho obsah.

Výběr ukládaných skriptů

Extraktor nabízí možnost nastavit, pro které typy objektů se ukládají jejich DDL.

Filtrování schémat

Lze nastavit, ze kterých schémat se mají objekty z databáze extrahovat, a to pomocí dvou filtrů. První filtr určuje schémata, která mají být zahrnuta v extrakci (include filter) a druhý filtr schémata, která mají být z extrakce vyloučena (exclude filter). Oba filtry se zadávají v podobě regulárního výrazu a mohou být nastaveny současně – v takovém případě se extrahují pouze objekty zahrnutých schémat, která nejsou vyloučena.

Filtrování systémových objektů

Nástroj nabízí možnost vyfiltrování systémových objektů z extrahovaných. V případě jejich zahrnutí nemusí být extrahována všechna metadata takových objektů (to se týká zejména DDL, která nebývají pro systémové objekty dostupná).

4.2 Omezení

Rozdíly v možnostech datové struktury projektu Manta oproti struktuře objektů v DB2 jsou následující:

- DB2 nerozlišuje u metod objektového typu, zdali jsou funkcemi nebo procedurami. Tato informace se doplní v závislosti na existenci návratové hodnoty metody.
- Cílová platforma Manta podporuje přetížení (overload) procedur i funkcí pouze, pokud mají rozdílný počet parametrů. V případě stejného počtu jsou přepsány. Toto neplatí v DB2, kde je možné mít více signatur, které se liší pouze typem parametrů. V takovém případě se signatura dříve uložená ve slovníku přepíše a uloží se poslední.
- DB2 na rozdíl od datové struktury nástroje Manta podporuje hierarchii (dědění) tabulek a pohledů. Všechny tabulky i pohledy se do slovníku přidají bez ohledu na hierarchii se všemi (i zděděnými) sloupci.
- Datový typ row, používaný databází DB2, je možné vytvořit na základě tabulky (jednotlivé sloupce tabulky jsou položkami tohoto datového typu). Tuto konstrukci slovník nepodporuje, proto jsou všechny položky takového datového typu kopírovány bez ohledu na jeho závislost na tabulce.

V režimu kompatibility Oracle lze definovat dva různé kurzory i přes to, že databáze DB2 podporuje pouze jeden typ kurzoru. Všechny kurzory extrahované z databáze se přidají do slovníku stejným způsobem.

4.3 Použité technologie

Nástroj bude implementován v programovacím jazyce Java s použitými frameworky *Spring* a *MyBatis*. Správu závislostí bude mít na starosti nástroj *Maven* a testy budou psány s pomocí frameworku *JUnit*.

Java, Spring a Maven

Programovací jazyk je vybrán s ohledem na snadnou integraci s ostatními nástroji Manta, stejně tak jako framework *Spring* a nástroj *Maven* pro správu závislostí. Ostatní části projektu Manta, se kterými se bude extraktor integrovat, jsou implementovány v jazyce Java s použitím frameworku *Spring* pro co největší nezávislost na platformě, na které běží. Všechny potřebné části (například metadatový slovník) se nacházejí v *Maven* repozitáři.

MyBatis Framework

Databázový framework *MyBatis* je vybrán pro jeho možnost přesného definování použitých SQL dotazů. Ty je možné mít uložené v souborech, kde je lze snadno upravit bez nutnosti zásahu do zdrojového kódu nástroje. To je možné využít například při požadavku upravit některý z SQL dotazů pro potřeby zákazníka.

JUnit Framework

Výběr testovacího frameworku *JUnit* je vhodný zejména pro jeho velice dobrou podporu ve *Spring* frameworku a ostatních nástrojích, jako je *Maven*.

4.4 Volba způsobů extrakce

Metadata objektů

Metadata všech objektů se budou extrahovat z pohledů databázového katalogu. Katalogové pohledy databáze DB2 nabízejí všechna potřebná metadata pro naplnění metadatového slovníku.

DDL

Primárně se budou SQL skripty extrahovat také z databázového katalogu. Objekty, které mají tuto možnost, jsou:

- pohledy,
- spouštěče,
- metody,
- funkce,
- procedury.

Pro objekty, které nemají DDL ve svém katalogovém pohledu, bude použita možnost vygenerování DDL pomocí systémové procedury. To se týká těchto objektů:

- schémata,
- proměnné,
- všechny uživatelské datové typy,
- moduly (to zahrnuje i Oracle balíčky),
- sekvence,
- tabulky.

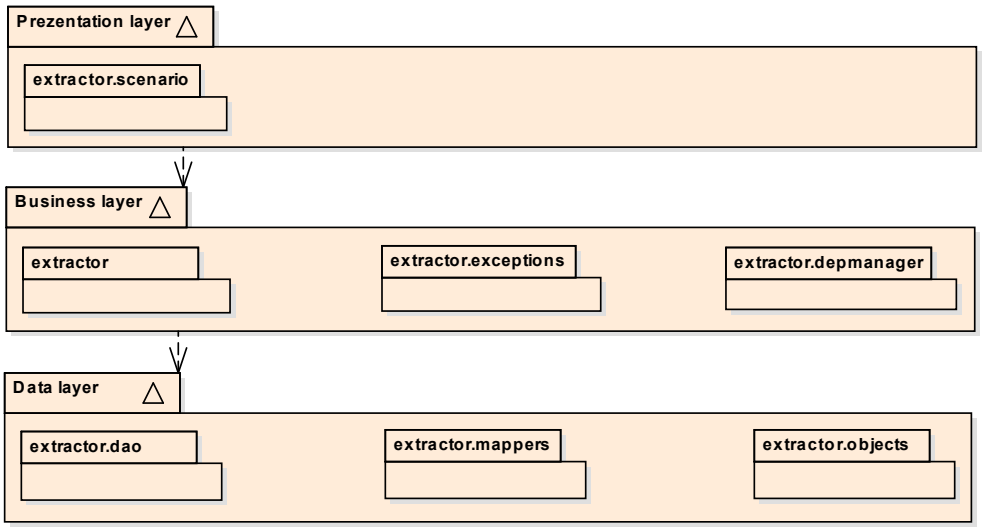
Nakonec objektům, které nemají možnost generování DDL ani pomocí systémové procedury, bude vygenerována základní verze jejich DDL přímo v extraktoru. Těmito objekty jsou servery a přezdívky.

Kapitola 5

Architektura

5.1 Logický pohled

Implementace extraktoru dodržuje třívrstvou architekturu. Prezentační vrstvou nástroje je balíček `scenario` poskytnutý týmem Manta společně s rozhraním extraktoru. Do byznysové vrstvy patří mimo jiné samotný extraktor, jmenovitě kořenový balíček `extractor`, a také implementace manažeru pro řazení podle závislostí (dependency manager) v balíčku `depmanager`. Poslední datová vrstva obsahuje třídy reprezentující databázové entity v balíčku `objects` společně s implementací tříd poskytujících přístup do databáze (balíčky `dao` a `mappers`. Pro úplný přehled balíčku viz obrázek 5.1.



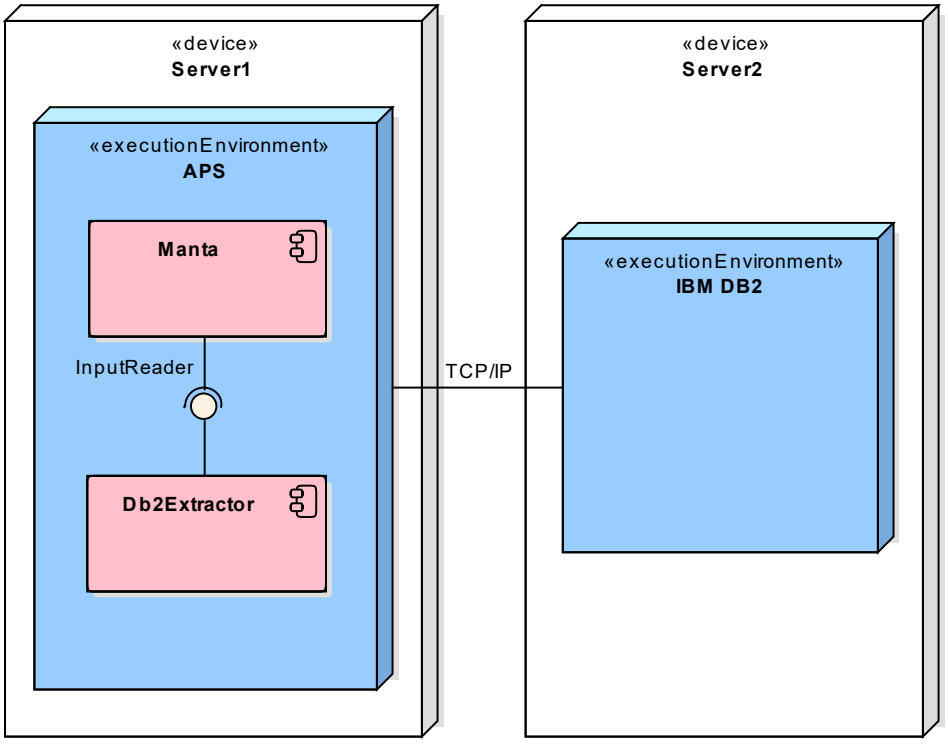
Obrázek 5.1: Diagram logického rozdělení balíčku

5.2 Implementační pohled

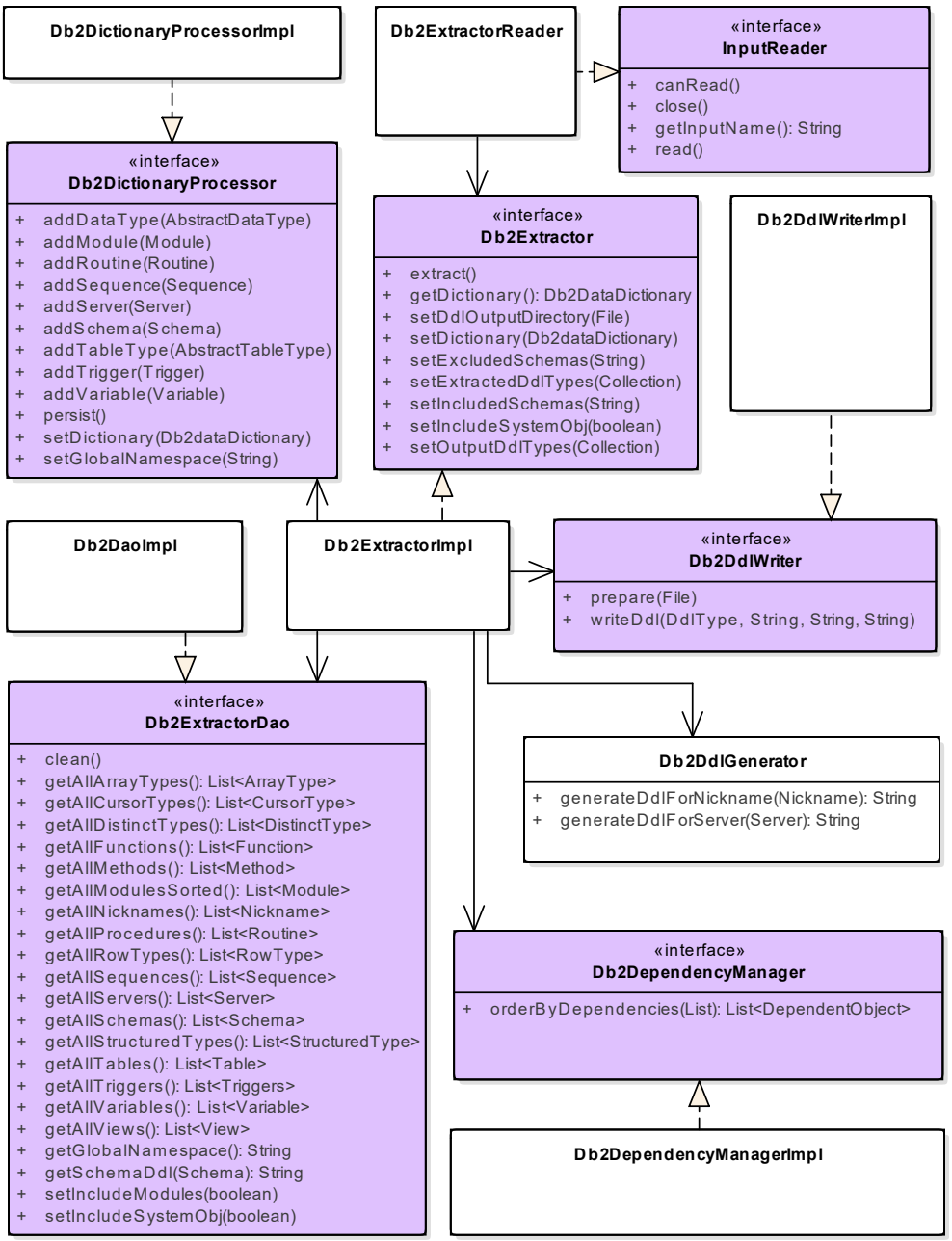
Implementační pohled zachycuje spolupráci hlavních komponent extraktoru (viz obrázek 5.3). Nejdůležitější komponentou nástroje je `Db2ExtractorImpl`, který implementuje rozhraní `Db2Extractor` a společně se třídami jako `Db2DdlGenerator` a `Db2DependencyManagerImpl` tvoří hlavní část jeho logiky. Pro detailní popis zodpovědností jednotlivých tříd nástroje viz kapitola 6.

5.3 Nasazení

Obrázek 5.2 ukazuje jednoduchý diagram typického nasazení extraktoru. Na zařízení pojmenovaném *Server1* běží aplikační server (*APS*), na kterém jsou spuštěny nástroje Manta (pro zjednodušení pouze jako jediná komponenta *Manta*). Společně s *Mantou* je nasazen extractor *Db2extractor*, se kterým *Manta* komunikuje přes rozhraní `InputReader`. Aplikační server není blíže specifikován, nástroje na něm nejsou závislé (díky použití frameworku *Spring*). Na druhém zařízení *Server2* je spuštěna instance databáze *IBM DB2*, se kterou extraktor komunikuje pomocí jejího webového rozhraní.



Obrázek 5.2: Diagram nasazení extraktoru



Obrázek 5.3: Implementační pohled na extraktor metadat

Kapitola 6

Detailní design

Kapitola popisuje jednotlivé třídy aplikace, jejich rozhraní a zodpovědnosti. Stěžejní části jsou popsány podrobněji. V kapitole nejsou popsány všechny konstrukce specifické pro použitý programovací jazyk.

6.1 Db2Extractor a Db2ExtractorImpl

Rozhraní `Db2Extractor` implementuje třída `Db2ExtractorImpl`. Rozhraní bylo poskytnuto týmem projektu Manta pro implementaci extraktoru a předepisuje několik základních metod, ze kterých mimo jiné vychází požadavky v kapitole 4.1. Předepsané metody se použijí převážně pro změnu výchozího nastavení extraktoru před jeho spuštěním. Metody předepsané rozhraním jsou následující:

- Pro nastavení adresáře, do kterého se budou zapisovat soubory s extrahovanými DDL, se použije metoda `setDdlOutputDirectory`.
- O nastavení typů databázových objektů, které budou extrahovány, se stará `setExtractedDdlTypes` a o ty, které budou ukládány, `setOutputDdlTypes`.
- Další je nastavení regulárních výrazů pro filtry pomocí `setIncludedSchemas` a `setExcludedSchemas`.
- Poslední možností nastavení je metoda `setIncludeSystemObj`, s jejíž pomocí je možné filtrovat systémové objekty z extrahovaných.
- Nakonec metoda `extract` spustí extrakci s výchozím, či změněným nastavením.

Třída `Db2ExtractorImpl` je stěžejní částí aplikace. Používá rozhraní `Db2ExtractorDao` pro přístup k datové vrstvě, `Db2DictionaryProcessor` pro zá-

pis získaných dat do metadatového slovníku a `Db2DdlWriter` k uložení extrahovaných skriptů na disk.

6.1.1 extract

Po zavolání metody `extract` se nejprve provede kontrola nastavení metadatového slovníku a poté nastavení ostatních tříd, se kterými `Db2ExtractorImpl` spolupracuje, tj. změna nastavení `Db2ExtractorDao`, předání slovníku pro `Db2DictionaryProcessor` a zavolání metody `prepare` na `Db2DdlWriter`. Jako poslední se nastaví do slovníku jméno globálního jmenného prostoru (global namespace).

Jakmile skončí všechna nastavení, začíná samotná extrakce. Další část textu se zabývá těmito skupinami objektů:

- schémata,
- servery,
- moduly,
- sekvence,
- spouštěče,
- metody,
- procedury,
- funkce.

Pro každou skupinu objektů existuje metoda `extract*` (například `extractModules`), která získá pomocí `Db2ExtractorDao` seznam objektů se všemi potřebnými a dostupnými informacemi. Pro každý objekt ze seznamu je volána příslušná metoda `add*` rozhraní `Db2DictionaryProcessor`, jehož implementace přidá objekt do slovníku, a nakonec pro celý seznam metoda `saveAllDdl`. Pořadí extrakce je zvoleno tak, aby nevnikaly problémy se závislostmi mezi objekty.

Pro každý typ objektů platí, že je extrahován v případě, že je mezi typy nastavenými pro extrakci. Pokud při získávání metadat z databáze nastane chyba a informace se nepodaří získat, skupina objektů tohoto typu se přeskočí.

Chování metody `extract*` je pro některé ze jmenovaných typů objektů rozšířeno o další akce:

- V případě `extractSchemas` jsou schémata po získání jejich seznamu filtrována pomocí `filterSchemas`. Schémata jsou extrahována bez informace o jejich DDL. Pro vygenerování jejich DDL se použije `generateAllSchemaDdls`. Nakonec je seznam vyfiltrovaných schémat vrácen pro další použití.

- Servery také nejsou extrahovány s jejich DDL. Tuto informaci není možné z databáze extrahovat, proto je jejich DDL vygenerováno pomocí statické metody `generateDdlForServer` nachazející se ve třídě `Db2DdlGenerator`.
- Extrakce metadat pro moduly je rozšířena o ukládání DDL jejich těl (package body) v případě, že se nejedná o standardní DB2 modul, ale jedná se o balíček Oracle. Pro ukládání druhého DDL je volána přímo metoda `writeDdl` přes rozhraní `Db2DdlWriter` a to pouze v případě, že mají být extrahována těla balíčků.
- Všem metodám `extract*`, kromě serverů a schémat, je předáván seznam schémat, ze kterých mají být objekty extrahovány.

Zbylé objekty se extrahují pomocí metody `extractObjectsWithDependencies`. Důvodem jsou možné závislosti mezi nimi. Objekt, který je závislý na jiném, musí být do slovníku přidán nejdříve po přidání všech objektů, na kterých závisí. Tato část se týká těchto skupin objektů:

- tabulky,
- pohledy,
- proměnné,
- přezdívky,
- uživatelem definované datové typy.

Postupně se projdou všechny výše zmíněné skupiny a pokud je skupina mezi extrahovanými typy, následuje získání seznamu všech objektů skupiny pomocí `Db2ExtractorDao`. Pomocí `saveAllDdl` jsou uložena DDL objektů v seznamu. Celý seznam získaných objektů je vložen do nového seznamu objektů, mezi kterými mohou být závislosti.

Jediný rozdíl je v případě získávání seznamu přezdívek. Přezdívky jsou extrahovány z databáze bez jejich DDL, protože stejně jako v případě serverů tuto informaci není možné z databáze extrahovat. Před uložením jejich DDL je pro každou přezdívku vygenerováno DDL pomocí `generateDdlForNickname` ve třídě `Db2DdlGenerator`.

Nakonec je celý seznam objektů s možnými závislostmi seřazen pomocí `orderByDependencies` třídy implementující rozhraní `Db2DependencyManager`. Každý objekt, který má být tímto způsobem řazen, musí splňovat rozhraní `DependentObject` (viz kapitola 6.5).

Jakmile jsou objekty seřazeny, následuje jejich postupné přidání do slovníku.

Po skončení extrakce všech objektů se volá metoda `persist` na `Db2DictionaryProcessor` pro uložení všech změn ve slovníku a také metoda

`clean` na `Db2ExtractorDao` pro vymazání všech použitých záznamů z pomocných tabulek.

6.1.2 saveAllDdl

Metoda má parametrem jakýkoli seznam objektů, které dědí `AbstractDb2Object` (viz kapitola 6.7). Tento předek poskytuje všechny potřebné informace pro uložení DDL, kterými jsou typ objektu, schéma, název DDL (ten je použit pro název souboru) a obsah DDL.

Pro každý objekt seznamu se kontroluje, zdali je mezi objekty, které mají být uloženy na disk. Samotné uložení DDL probíhá zavoláním `writeDdl` přes rozhraní `Db2DdlWriter`.

6.1.3 filterSchemas a includeExcludeSchema

Zde probíhá filtrování schémat. Každý název schématu je kontrolován pomocí `includeExcludeSchema`, zda vyhovuje nastaveným filtrům.

Pokud nastavením nejsou zahrnuty systémové objekty (tedy ani systémová schémata) a filtr zahrnutých schémat (include) je prázdný, tak se odfiltrují systémová schémata. Toto neplatí pro schéma *SYSPULIC*, do tohoto schématu jsou ukládány všechny veřejné objekty, proto není vyfiltrováno. V opačném případě neprobíhá filtrování systémových schémat.

Ostatní schémata jsou vyfiltrována standardním filtrem, který používá *Manta*, tj. metodou `includeExclude` třídy `SchemaFilter`, které se předají nastavené filtry. V případě, že nebyla předem filtrována systémová schémata, stále mohou být vyfiltrována tímto filtrem.

6.1.4 generateAllSchemaDdls

Z důvodu obrovské náročnosti generování DDL pro schémata (v databázi se pomocí procedury *db2lk_generate_ddl* společně s jeho DDL generují DDL pro všechny objekty ve schématu, viz metoda `callDb2lkGenerateDdl` 6.2.1) se generují DDL pouze pro vyfiltrovaná schémata.

Pokud jsou schémata mezi typy, kterým mají být ukládána DDL, potom pro každé schéma uloží jeho DDL s pomocí `getSchemaDdl`, kterou nabízí `Db2ExtractorDao`.

6.2 Db2ExtractorDao a Db2DaoImpl

`Db2DaoImpl` implementuje rozhraní `Db2ExtractorDao` a poskytuje přístup k informacím uloženým v databázi DB2. Přístup k datům je zajištěn pomocí rozhraní, která implementuje framework *MyBatis* (zvaných „mappers“) na základě souborů, ve kterých jsou definovány dotazy do databáze. Tato rozhraní se nachází v balíčku `extractor.mappers`. Kapitola nepopisuje fungo-

vání frameworku. Použité pohledy a tabulky, ze kterých jsou data vybírána, jsou popsány v kapitole 3.

Nastavení

Chování třídy lze nastavit několika různými způsoby. Nastavení `setIncludeModules` říká, zdali mají být do výsledků zahrnuty i objekty, které jsou definovány v modulech, a `setIncludeSystemObj` řeší zahrnutí systémových objektů. Posledním nastavením je zahrnutí či vyloučení synonym pomocí `setIncludeSynonyms`. Nastavené hodnoty budou použity jako parametry při skládání dotazů do databáze a to tak, že podle nastavené hodnoty bude výsledek obsahovat takové objekty či nikoli.

6.2.1 callDb2lkGenerateDdl

Metoda volá v databázi na systémovou proceduru `db2lk_generate_ddl`. Tato procedura vygeneruje DDL podle parametrů zadaných na vstupu a naplní jimi tabulku `systools.db2look_info`. Procedurou je vrácen identifikátor vytvořených záznamů (tzv. „token“), ten je dále použit při výběru DDL ze zmíněné tabulky.

Omezením procedury je, že ji lze použít pouze pro vygenerování DDL pro samostatnou tabulku, všechny objekty ve schématu nebo celou databázi. Není známa možnost, jak jednotlivě generovat DDL pro různé typy objektů. Pomocná tabulka pro vygenerovaná DDL není dočasnou tabulkou a není automaticky promazávána, proto je nutné použité záznamy z tabulky po skončení vymazat (více viz metoda 6.2.2).

6.2.2 clean

Pomocí `clean` je zajištěno mazání již nepotřebných záznamů v pomocné tabulce `systools.db2look_info`. Mazání se provádí pomocí volání databázové procedury `db2lk_clean_table`, která jako parametr vyžaduje token vygenerovaný pomocí `db2lk_generate_ddl` (více viz metoda 6.2.1). Poté z tabulky smaže záznamy s tímto identifikátorem.

Implementace poskytuje dvě možnosti přístupu k mazání záznamů z pomocné tabulky. Při zavolání `clean` se provede jedna z následujících metod:

- `cleanUsedTokensOnly` pomocí metody `callDb2lkCleanTable` volá databázovou proceduru pro všechny tokeny vygenerované při běhu extraktoru,
- `cleanEntireDb2lookInfoTable` pracuje stejným způsobem jako předchozí s tím rozdílem, že nejprve získá seznam tokenů všech záznamů v pomocné tabulce.

Mezi těmito možnostmi lze přepínat pomocí `setCleanEntireDb2lookInfo-Table`.

6.2.3 getDb2lkTokenForSchema

Tato metoda zjistí identifikátor (token) pro DDL všech objektů zadaného schématu v pomocné tabulce `systools.db2look_info`. Drží si seznam vygenerovaných tokenů pro schémata. Pokud token hledaného schématu není známý, pomocí `callDb2lkGenerateDdl` zavolá databázovou proceduru pro jeho vygenerování. Informace o tokenu i volání procedury pro hledané schéma se uloží, čímž je zajištěno, že se nebudou DDL generovat vícekrát pro jedno schéma.

6.2.4 getGlobalNamespaceName

Získá jméno pro metadatový slovník. Jméno se skládá ze tří částí: jméno hostitelského systému, instance a databáze. Tyto části se získají z databáze a pomocná třída `Db2NameUtils` se postará o správný tvar výsledného jména.

6.2.5 getAll*

Metody s prefixem `getAll*` (například `getAllMethods`) mají za úkol získat z databáze všechny objekty některého z typů se všemi potřebnými a dostupnými informacemi. Pro rozdělení skupin objektů podle toho, jak se informace získají, viz kapitola 4.4. Názvy všech objektů jsou ořezány zprava o bílé znaky.

Pro objekty, které mají všechny informace uloženy ve svém katalogovém pohledu, se provede pouze jednoduchý výběr z databáze. Stejně tak u takových, pro které nelze DDL vybrat nebo generovat v databázi (více o získání jejich DDL viz kapitola 6.6).

U ostatních objektů se pomocí metod `get*Ddl` získá token vygenerovaných DDL pro celé schéma voláním `getDb2lkTokenForSchema`. Pokud se nepodaří DDL vygenerovat, například běhovou chybou procedury, nelze již tato DDL získat (to neplatí v případě tabulek a Oracle balíčků, viz dále). V opačném případě se provede jedna z následujících akcí:

- V případě DDL pro schémata a sekvence se vybere výsledek z pomocné tabulky `systools.db2look_info`.
- U proměnných a všech uživatelsky definovaných typů je nutné rozlišit, zdali se nacházejí ve schématu nebo modulu:
  - DDL objektů nacházejících se ve schématu se vyberou stejným způsobem jako v předchozím případě.
  - U objektů v modulech je situace složitější. Pomocná tabulka nemá sloupec s informací o tom, v jakém modulu se objekt nachází, pouze jeho přesný čas vytvoření (timestamp), který se shoduje s časem

vytvoření uloženým v katalogových pohledech. V tomto případě je tedy vrácen seznam stejnojmenných objektů, které se nacházejí v modulech, jsou jednoho typu a ve stejném schématu. Z tohoto seznamu se vybere výsledek podle času vytvoření a ověří se, zdali DDL obsahuje jméno modulu. Každé takové DDL vzniká jako příkaz na změnu (ALTER) modulu. Na základě testů se nelze odkazovat z jednoho modulu na objekty v jiném modulu (to znamená, že takové DDL neobsahuje jméno jiného modulu). Také se při testování nepodařilo vytvořit konfliktní objekty v naprosto stejný čas.

- DDL tabulek jsou také vybírána z pomocné tabulky. Případy se liší v použitém tokenu. Když se nepodařilo vygenerování tokenu pro objekty v celém schématu, je vygenerováno DDL a k němu příslušný token pouze pro tabulku.

Kurzory jsou z databáze extrahovány společně s Oracle kurzory, které jsou reprezentovány jako proměnné. Tyto proměnné jsou při výběru proměnných z databáze vyfiltrovány.

Přístup k DDL modulů se liší v případě modulů DB2 a Oracle balíčků. Před pokusem o získání tokenu nejprve metoda `getAndSaveModuleDdl` zjistí, o který typ modulu se jedná, a poté se zavolá `getAndSaveDdlForDb2Module-OrAlias` nebo `getAndSaveDdlForOraclePackage`.

V případě klasických DB2 modulů se pouze vybere výsledek z pomocné tabulky a takto získané DDL se nastaví modulu. Pokud nelze vygenerovat token, není možné DDL získat.

U Oracle balíčků se situace liší. Nejprve se klasickým způsobem pokusí získat token. Při jeho úspěšném získání se z pomocné tabulky vybere seznam DDL balíčku (tj. DDL package a package body). Tělo balíčku vždy následuje v seznamu po definici balíčku. Pokud token nebyl získán, obě DDL se vyberou ze systémové tabulky `sysibm.sysmodules`. Postup s výběrem ze systémové tabulky není zvolen jako primární ani jako jediný. Je to z důvodu zachování jednotného přístupu ke generování DDL a také kvůli běžným a častým změnám systémových tabulek v různých verzích databází.

6.3 Db2DictionaryProcessor a Db2DictionaryProcessorImpl

Třída `Db2DictionaryProcessorImpl` implementuje rozhraní `Db2Dictiona-ryProcessor` pro práci s databázovým slovníkem. Objekty musí být do slovníku přidávány ve správném pořadí. Pokud není nalezen některý z objektů, na kterém přidávaný objekt závisí, není ho možné přidat. Na konci práce se slovníkem je nutné zavolat metodu `persist` pro uložení provedených změn.

6.3.1 add*

Analogicky jako v předchozích třídách je zde sada metod začínajících prefixem `add*` (například `addSchema`). Na začátku se vždy zjistí uzel, který zastupuje rodiče, kterému bude přidán nový potomek pomocí `getParent`. Toto neplatí u serverů a schémat, ta jsou přidána automaticky do kořenového uzle. Pro hledání datových typů ve slovníku se použije metoda `getDataType`.

Objekty jsou do slovníku přidávány pomocí připravených metod třídy `Db2DataDictionary`. Práce s těmito metodami nebude dále popsána. Každý objekt ve slovníku má seznam svých vlastností. Následuje stručný popis toho, s jakými vlastnostmi jsou jednotlivé typy objektů přidány:

- Proměnné, schémata, pole, objektové typy, servery, spouštěče, sekvence, tabulky a pohledy mají ve slovníku své charakteristické vlastnosti, se kterými jsou tedy i přidány.
- Datový typ distinct je do slovníku přidán se stejnými vlastnostmi jako datový typ, který reprezentuje, mimo vlastnost „vestavěný datový typ“.
- Datový typ row je přidán jako datový typ s vlastností `RECORD`. Jednotlivé položky tohoto typu jsou přidány pomocí metody `addTypeField` s vlastností `COMPOSITE ITEM`.
- Všechny kurzory v aplikaci jsou přidány s vlastností `REF CURSOR TYPE`.
- Moduly i balíčky jsou do slovníku přidány s vlastností `PACKAGE`, není tedy rozlišeno, o jaký typ modulu se jedná.
- Přezdívky se do slovníku přidávají jako pohledy, tedy s vlastností `VIEW`.
- Metody se vloží do slovníku jako funkce nebo procedury, tedy s vlastností `FUNCTION` nebo `PROCEDURE` podle toho, jestli existuje jejich návratová hodnota.

Dále následuje popis několika důležitých implementačních detailů:

- Před přidáním nového datového typu pomocí `addDataType` se kontroluje, zdali tento typ už není ve slovníku. V případě jeho přítomnosti se znovu nepřidává.
- Pole mohou mít libovolný datový typ hodnot i datový typ indexu. Pokud není datový typ indexu specifikován, jako výchozí typ je nastaven `INTEGER`. Datovým typem hodnot pole může být mimo jiné datový typ tabulky. Pokud tomu tak je, atribut určující datový typ hodnot pole je `ROW` a název tabulky se získá z atributů `tableRowSchemaName` a `tableRowName`.

- Datové typy jednotlivých položek a sloupců mohou být referencemi (datový typ `REFERENCE`). V takovém případě se datový typ, na který odkazuje reference, bere z atributů s prefixem `refType*`.
- Hodnota kurzoru může být kterýkoli datový typ, také datový typ tabulky nebo proměnné. O který typ se jedná, se pozná z atributu kurzoru `depType`.
- Pokud není ve slovníku nalezen předek objektového typu, je takový typ přesto přidán do slovníku.
- Funkce mohou mít jako svoji návratovou hodnotu datový typ nebo tabulku. V případě tabulky je vytvořen datový typ tabulky jako návratová hodnota funkce. Funkce, která má jako návratovou hodnotu tabulku, nemá v katalogu definovaný návratový typ. Sloupce návratové tabulky jsou mezi parametry funkce.
- Mezi parametry metod je v některých případech přidán parametr s názvem `SELF`. Tento parametr má datový typ shodný s objektovým typem, ke kterému je metoda definována. Takový parametr nepatří do předpisu metody a nelze ho v DDL definovat. Proto není přidán mezi parametry ve slovníku.
- Některé parametry nemají v databázovém katalogu své jméno. Jména těchto parametrů jsou nahrazena jejich pořadovým číslem s prefixem „$“.

6.3.2 getDataType

Pro získání datových typů ze slovníku se využívá této jediné metody. Pokud se jedná o vestavěný datový typ, název schématu je `SYSIBM`. Na základě toho lze rozhodnout, zdali hledat mezi vestavěnými nebo uživatelskými datovými typy.

6.4 Db2DdlWriter a Db2DdlWriterImpl

Třída implementující rozhraní `Db2DdlWriter` se používá pro vytváření adresářové struktury a zápis DDL do souborů.

6.4.1 prepare

Tato metoda musí být zavolána před začátkem ukládání DDL. Jejím úkolem je zejména připravit kořenový výstupní adresář. Kontroluje nastavení adresáře a jeho existenci. V případě, že adresář neexistuje, vytvoří ho a v opačném případě obsah adresáře smaže.

6.4.2 writeDdl

Při zavolání této metody se nejprve pomocí nastaveného příznaku zkontroluje, jestli byla volána metoda `prepare` a výstupní adresář je tedy v pořádku. Poté se získá cílový adresář pro uložení souboru pomocí metody `getDirectory`, která buď adresář nalezne, nebo ho vytvoří. Adresářová struktura je následující: kořenový adresář, adresář s normalizovaným názvem schématu (pokud se objekt nachází ve schématu) a nakonec adresář s názvem typu objektu. Soubor je uložen pod normalizovaným názvem, který je předán parametrem. Normalizace jmen se provádí pomocí `normalizeFilename`.

6.4.3 normalizeFilename

Názvy objektů v databázích mohou mít znaky, které nepovoluje operační systém. Také je možné mít více názvů různých objektů, které jsou z pohledu operačního systému stejné. Toto řeší `normalizeFilename` tak, že nahradí všechny speciální (nepovolené) znaky názvů za podtržítka. Tím ale vznikají další kolizní názvy. `Db2DdlWriter` má mapu originálních jmen a jim přidělených čísel. Vždy se nejdříve zjistí, jestli už je jméno používáno a v takovém případě se použije jemu přiřazené číslo. Pokud jméno nemá své číslo, získá se z něj co nejvíce kolizní jméno (odstraněním nepovolených znaků a převedením na malá písmena). V mapě kolizních jmen se získá informace o tom, kolikátou kolizí toto jméno je, a toto číslo se uloží k původnímu jménu.

Metoda vrací normalizovaná jména a vytváří koncovku jmen „.1“, „.2“ atd. podle pořadí kolize.

6.5 Db2DependecyManager a Db2DependencyManagerImpl

Implementace tohoto manažeru se používá k řazení objektů podle jejich závislostí. V seřazeném seznamu jsou nejprve objekty bez závislostí a za nimi následují objekty, které na nich mají závislost. Řazení obstarává jediná metoda `orderByDependencies`. Každý objekt, který chce být řazen pomocí tohoto manažeru, musí splňovat rozhraní `DependentObject`. Řazení poté probíhá s pomocí grafu závislostí.

Po zavolání metody pro řazení se nejprve připraví struktura grafu v metodě `prepare`. Projde se seznam vstupních objektů a pro každý objekt se vygenerují jeho závislosti pomocí metody `generateDependencies` (viz níže), kterou implementuje každý z objektů se závislostmi přes rozhraní `DependentObject`. Podle získaných závislostí objektu se provede jedna z následujících akcí:

- Pokud objekt nemá žádné závislosti, je zařazen do vznikajícího grafu závislostí a také do pole seřazených objektů.

- Objekt se závislostmi je zabalen pomocí instance třídy `DependentObjectWrapper`, kde se k objektu přidají jeho závislosti. Takto zabalený objekt se přidá do seznamu objektů určených k řazení (`notSortedList`).

Objekt je přidán do grafu závislostí pomocí metody `add`. Nejprve se najde předek nového uzlu. Tím je objekt typu `DependentObjectHolder`, který reprezentuje buď schéma nebo modul. Pokud takový předek v grafu neexistuje, vytvoří se.

Tento předek drží mapu potomků uzlu roztřízených do jmenných prostorů, které jsou rozděleny podle `DependentObjectType`.

Jakmile je příprava hotova, začíná samotné řazení pomocí metody `order`. Řazení probíhá ve smyčce, dokud není prázdný seznam objektů určených k řazení. U každého objektu se pomocí metody `removeSatisfiedDependencies` projde také seznam jeho závislostí a každá splněná závislost se ze seznamu smaže. Závislost je splněna, pokud se objekt, který je předmětem závislosti, nachází v grafu závislostí. V případě, že je na konci seznam závislostí prázdný, objekt je přidán do grafu, do seznamu sežazených a odstraněn ze seznamu k řazení.

Řazení se také zastaví, pokud se při průchodu celým seznamem žádný objekt nepřidá do grafu. V takovém případě jsou zbylé objekty ze seznamu neseřazených přesunuty na konec seznamu seřazených. Nakonec je vrácen seznam seřazených objektů.

Rozhraní DependentObject

Toto rozhraní musí splňovat každý objekt, který chce být řazen. Rozhraní definuje metody, pomocí kterých lze určit předka uzlu v grafu, a také metodu `generateDependencies`. Ta předepisuje vygenerování seznamu závislostí objektu. Seznam je reprezentován jako spojový (linked list). Důvodem je časté mazání ze seznamu. Pomocí metody `getDependentObjectType` objekt vrátí informaci o tom, do jakého jmenného prostoru patří.

Rozdělení podle DependentObjectType

`DependentObjectType` rozděluje jmenné prostory v grafu závislostí. V databázi je možné mít více stejných názvů v jednom schématu nebo modulu pro odlišné typy objektů. Přesněji jsou to: uživatelem definované typy, všechny tabulkové typy a proměnné. Jmenných prostor je v databázi více, výčet se týká pouze objektů se závislostmi.

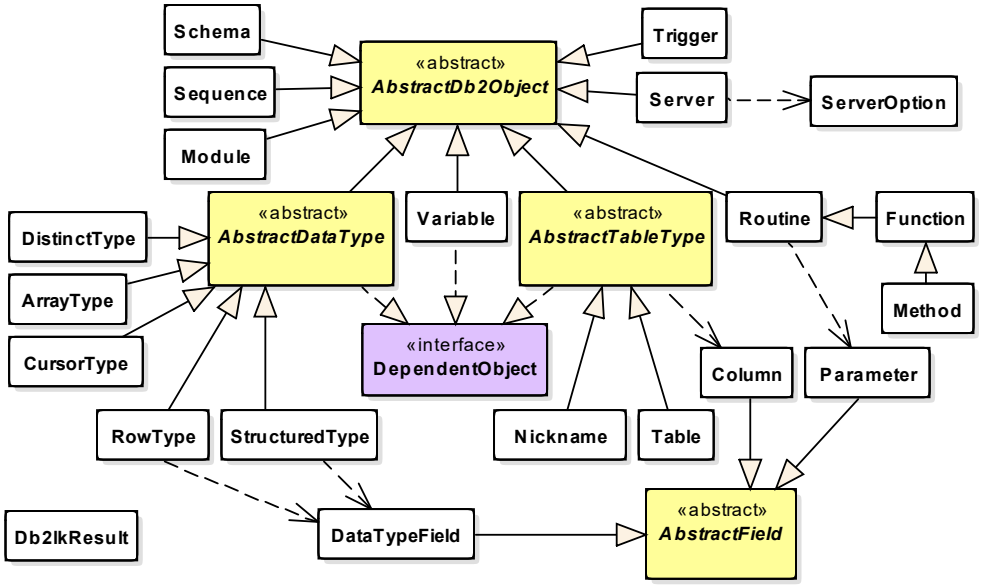
6.6 Db2DdlGenerator

Tato třída je použita ke generování DDL objektů, pro které není možné získat DDL z databáze. Na začátek vygenerovaného skriptu je umístěno varování,

že skript byl generován. DDL pro server je tvořeno na základě katalogových pohledů SERVERS a SERVEROPTIONS, bohužel některé záznamy v SERVEROPTIONS se nemohou nacházet v části OPTIONS DDL příkazu, proto jsou filtrovány pomocí připraveného seznamu. Dále lze vygenerovat DDL pro přezdívku, která je vytvořena na základě vzdálené tabulky. Pro jiný typ přezdívky (například vytvořené na základě strukturovaného souboru) se nepodařilo nalézt všechna potřebná metadata. Takový DDL příkaz nebude vygenerován.

6.7 Objekty v extraktoru

Objekty používané v extraktoru mají svojí hierarchii, ta je zachycena na obrázku 6.1.



Obrázek 6.1: Objekty používané extraktorem

Kapitola 7

Testování

Implementaci extraktoru pokrývají standardní jednotkové testy (unit tests). Dále jsou připraveny dva integrační testy. Každý integrační test se spouští proti databázi, na kterou je nahrán připravený SQL skript pro vytvoření testovaných objektů.

7.1 Test DB2 databáze

Text dále popisuje spuštění integračního testu proti databázi vytvořené za pomoci připraveného skriptu. Pro spuštění testu je nutné provést následující kroky:

1. Vytvořit databázi pomocí příkazu `CREATE DATABASE TESTDB`. Tento příkaz se spouští v příkazovém řádku databáze. Pro jeho inicializaci je nutné zavolat příkazy `db2cmd` a následně `db2`

2. Nad vytvořenou databází spustit připravený skript `testDB.sql` přibalený k aplikaci v adresáři „src/test/resouces“.

3. Změnit adresu, uživatelské jméno a heslo v souboru `TestDb.properties`, který se nachází ve stejném adresáři jako skript.

4. Odstranit anotaci `@Ignore`, která je definována ve třídě s testem `ExtractorDb2Test`.

7.2 Test databáze v režimu kompatibility

Následuje integrační test proti databázi v režimu kompatibility Oracle, která je vytvořena za pomoci připraveného skriptu. Postup je téměř totožný s předchozím.

1. Před vytvořením databáze je nutné ji přepnout do režimu kompatibility. Bez tohoto kroku nelze pokračovat. Přepnutí režimu databáze se provádí v příkazovém řádku. Pro zapnutí režimu kompatibility slouží tyto kroky:
   a) Nejprve se spustí databáze pomocí `db2start`.
   b) Dále se zavolji příkazy `db2set DB2_COMPATIBILITY_VECTOR=ORA` a `db2set DB2_DEFERRED_PREPARE_SEMANTICS=YES`.
   c) Databázi je nyní nutné restartovat a to sekvencí příkazů `db2stop` a `db2start`.
2. Nyní se vytvoří databáze příkazem `CREATE DATABASE TESTORA`.
3. Nad vytvořenou databází se spustí připravený skript `testOra.sql` přibalený k aplikaci v adresáři „src/test/resouces“.
4. Následuje změna adresy, uživatelského jména a hesla v souboru `TestOra.properties`, který se nachází ve stejném adresáři jako skript.
5. Na závěr se odstraní anotace `@Ignore`, která je definována ve třídě s testem `ExtractorOraTest`.

7.3 Test vytvoření serveru

Poslední test je součástí testu z kapitoly 7.1. Lze ho spustit pouze v případě, že jsou vytvořené obě databáze z integračních testů.

Test ověřuje přidání serveru a přezdívky. Pro spuštění testu je nutné provést následující kroky:

1. Upravit skript `testServer.sql` z adresáře „src/test/resouces“ doplněním uživatelského jména a hesla na dvě vyznačená místa ve skriptu.
2. Nad databází TESTDB spustit tento skript.
3. Odstranit anotaci `@Ignore` pro testovou metodu `testServer` v testovací třídě `ExtractorDb2Test`.

Kapitola 8

Uživatelská příručka

Upozornění: Zdrojové kódy tohoto nástroje mají závislosti na některých zdrojových kódech projektu *Manta*. Pokud je tedy zamýšleno použít nástroj v prostředí bez těchto závislostí, je nutná úprava zdrojových kódů, odstranění těchto několika závislostí a implementace nové třídy pro práci se slovníkem.

Manuál popisuje přidání extraktoru do existujícího projektu používajícího framework *Spring*, jeho nastavení a spuštění. Extraktor je přidán jako knihovna typu „jar“ (Java Archive). Postup je následující:

1. (Volitelné) Nejprve se zkompilují zdrojové kódy nástroje. Toto lze provést pomocí nástroje *Maven* příkazem `mvn clean install`.

2. Dále se do projektu přidá závislost na extraktoru. Toho lze docílit pomocí sestavovacího skriptu použitého nástroje pro správu závislostí, například *Maven* aj.

3. Jakmile je v projektu přidána závislost, do některého z aktivních nastavení frameworku *Spring* se přidá záznam `include` nastavení extraktoru `Db2Extractor.xml`.

4. Poté je potřeba v tomto nastavení definovat datový zdroj. To se provede přidáním záznamu `bean` s identifikátorem (id) „dataSource“. V této konfiguraci je upřesněna adresa databáze, uživatelské jméno a heslo. Příklad tohoto nastavení je `DatasourceTest` umístěný mezi testovými zdroji.

5. (Volitelné) Lze změnit nastavení promazávání pomocné tabulky pro generování DDL a to v konfiguraci nástroje `Db2Extractor.xml` v záznamu `bean` s identifikátorem „db2DaoImpl“.

6. Nyní jsou všechna nastavení hotová, je možné použít rozhraní extaktoru `Db2Extraktor` a případně změnit jeho výchozí nastavení pomocí metod s prefixem `set*`.

7. Nakonec se spustí extraktor zavoláním na metodu `extract`.

Závěr

Cílem práce bylo popsat strukturu metadat databáze IBM DB2, porovnat ji se dvěma zástupci konkurenčních databází a zjistit jejich dostupnost a způsoby extrakce. Mezi další cíle patřila volba nejlepšího ze způsobů extrakce a návrh a vytvoření prototypového řešení nástroje na extrakci těchto metadat pro datovou strukturu projektu Manta. Posledním neméně důležitým cílem bylo tento nástroj otestovat a vytvořit uživatelskou příručku.

V práci se podařilo dosáhnout všech vytyčených cílů. Byl nalezen způsob pro extrakci všech důležitých metadat a téměř všech skriptů. Pro jejich extrakci bylo nalezeno několik různých možností, ze kterých se vybralo nejlepší možné řešení. Dále se podařilo navrhnout, implementovat a otestovat nástroj pro extrakci metadat. Testy pokrývají nejdůležitější části implementovaného nástroje.

Kromě integrace tohoto nástroje do projeku Manta se do budoucna z jeho pohledu neočekávají žádné velké změny. V případě, že by došlo k rozšíření možností datové struktury projektu Manta, bylo by možné zbavit se některých omezení. Jinak nástroj čeká pouze standardní údržba či drobná úprava pro možné nové verze již zmíněné databáze.

Literatura

[1] *Manta Flow Demo*. 26.2.2017. Dostupné z: https://getmanta.com/manta-flow-demo/

[2] Quast, K.; Valenta, M.: Grafová databáze jako úložiště metadat pro data lineage – zkušenosti a výzvy. In *11th Workshop on Intelligent and Knowledge Oriented Technologies 35th Conference on Data and Knowledge*, STU Bratislava, 2016, ISBN 978-80-227-4619-9, str. 333.

[3] Liu, L.; Özsu, M. T. (editoři): *Encyclopedia of Database Systems*. Springer US, 2009, ISBN 978-0-387-35544-3.

[4] *Oracle9i Database Reference Release 2 (9.2) - The Data Dictionary*. [cit. 12.3.2017]. Dostupné z: http://docs.oracle.com/cd/B10501_01/server.920/a96524/c05dicti.htm

[5] *Microsoft SQL Server Language Reference - Information Schema Views*. [cit. 21.3.2017]. Dostupné z: https://msdn.microsoft.com/en-us/library/ms186778.aspx

[6] *Microsoft SQL Server Language Reference - Catalog Views*. [cit. 21.3.2017]. Dostupné z: https://msdn.microsoft.com/en-us/library/ms174365.aspx

[7] *IBM Knowledge Center - Compatibility features for Oracle*. [cit. 26.3.2017]. Dostupné z: https://www.ibm.com/support/knowledgecenter/SSEPGG_11.1.0/com.ibm.db2.luw.apdv.porting.doc/doc/c_compat_oracle.html

[8] *Dbeaver fórum, DB2 Generate DDL*. 15.4.2017. Dostupné z: http://dbeaver.jkiss.org/forum/viewtopic.php?f=2&t=1138

Příloha **A**

Seznam použitých zkratek

IBM DB2 LUW IBM DB2 for Linux, UNIX and Windows

DDL Data Definition Language

TCP/IP Transmission Control Protocol/Internet Protocol

Příloha B

Databázová práva

Příloha popisuje databázová práva přidělovaná v databázi DB2 pro spuštění extraktoru.

B.1 Zjištění práv uživatele

Pro zjištění aktuálních práv uživatele lze použít SQL dotaz:

- `SELECT AUTHORITY, D_USER, D_GROUP, D_PUBLIC, ROLE_USER, ROLE_GROUP, ROLE_PUBLIC, D_ROLE FROM TABLE (SYSPROC.AUTH_LIST_AUTHORITIES_FOR_AUTHID ('<jméno uživatele>', 'U') ) AS T ORDER BY AUTHORITY`

Zjištění práv skupiny lze provést stejným příkazem, kde se namísto `'U'` použije `'G'` a název skupiny.

B.2 Práva ve vytvořených databázích

Pokud je databáze vytvořená pomocí klíčového slova `restrict`, nejsou ve vytvářené databázi přidělena žádná práva. V opačném případě (standardně) jsou systémovému katalogu přidělena veřejná práva (public).

B.3 Minimální práva pro extraktor

Výčet minimálních práv v databázi vytvořené jako `restrict` pro spuštění extraktoru je následující:

- přidání práv na připojení do DB,
  - `GRANT CONNECT ON DATABASE TO USER <jméno uživatele>`
- nastavení pracovní zátěže (zde pouze základní),

  - `GRANT USAGE ON WORKLOAD SYSDEFAULTUSERWORKLOAD TO USER <jméno uživatele>`

- přidělení práv na balíčku používaném klientem,

  - `GRANT EXECUTE ON PACKAGE <název balíčku> TO USER <jméno uživatele>`

- práva na systémové procedury,

  - `GRANT EXECUTE ON PROCEDURE SYSPROC.DB2LK_GENERATE_DDL TO USER <jméno uživatele>`
  - `GRANT EXECUTE ON PROCEDURE SYSPROC.DB2LK_CLEAN_TABLE TO USER <jméno uživatele>`
  - `GRANT USAGE ON SEQUENCE SYSTOOLS.DB2LOOK_TOKEN TO USER <jméno uživatele>`

- přidělení práv pro práci s pomocnou tabulkou,

  - `GRANT ALL ON TABLE SYSTOOLS.DB2LOOK_INFO TO USER <jméno uživatele>`

- nastavení práv pro čtení ze systémových tabulek, které používá procedura, pomocí příkazů `GRANT SELECT ON <jméno tabulky> TO USER <jméno uživatele>` pro tyto tabulky:

```
SYSIBM.ROUTINES             SYSIBM.SYSSCHEMATA
SYSIBM.SYSDATATYPES         SYSIBM.SYSSERVERS
SYSIBM.SYSDEPENDENCIES      SYSIBM.SYSTABLES
SYSIBM.SYSDUMMY1            SYSIBM.SYSTABOPTIONS
SYSIBM.SYSFUNCMAPPINGS      SYSIBM.SYSTYPEMAPPINGS
SYSIBM.SYSFUNCTIONS         SYSIBM.SYSUSEROPTIONS
SYSIBM.SYSINDEXES           SYSIBM.SYSVARIABLES
SYSIBM.SYSINDEXOPTIONS      SYSIBM.SYSWRAPPERS
SYSIBM.SYSMODULES           SYSIBMADM.ENV_INST_INFO
SYSIBM.SYSROUTINEOPTIONS    SYSIBMADM.ENV_SYS_INFO
SYSIBM.SYSROUTINEPARMS      SYSTOOLS.DB2LOOK_INFO_V
SYSIBM.SYSROUTINES
```

- práva na jednotlivé katalogové pohledy, které používá buď procedura, nebo extraktor:

```
SYSCAT.ATTRIBUTES          SYSCAT.ROLES
SYSCAT.AUDITPOLICIES       SYSCAT.ROUTINEDEP
SYSCAT.CHECKS              SYSCAT.ROUTINEPARMS
SYSCAT.COLUMNS             SYSCAT.ROUTINES
SYSCAT.CONDITIONS          SYSCAT.ROWFIELDS
SYSCAT.CONTEXTS            SYSCAT.SCHEMATA
SYSCAT.CONTROLS            SYSCAT.SECURITYLABELCOMPONENTS
SYSCAT.DATATYPEDEP         SYSCAT.SECURITYLABELS
SYSCAT.DATATYPES           SYSCAT.SECURITYPOLICIES
SYSCAT.FUNCTIONS           SYSCAT.SEQUENCES
SYSCAT.HIERARCHIES         SYSCAT.SERVEROPTIONS
SYSCAT.INDEXES             SYSCAT.SERVERS
SYSCAT.INDEXEXTENSIONS     SYSCAT.STATEMENTS
SYSCAT.INDEXXMLPATTERNS    SYSCAT.TABCONST
SYSCAT.MODULEOBJECTS       SYSCAT.TABLES
SYSCAT.MODULES             SYSCAT.TABOPTIONS
SYSCAT.NICKNAMES           SYSCAT.TRIGGERS
SYSCAT.PACKAGES            SYSCAT.VARIABLES
SYSCAT.PERIODS             SYSCAT.VIEWDEP
SYSCAT.PROCEDURES          SYSCAT.VIEWS
```

Poznámka: Proceduru `db2lk` je nutné alespoň jednou zavolat z administrátorského účtu. Důvodem je vytvoření schématu `SYSTOOLS`, které nemusí být mezi existujícími schématy a při volání uživatelem s omezenými právy se nevytvoří.

Příloha C

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `impl` ..... zdrojové kódy nástroje i testů
  - `db2-dictionary-extractor` ..... extraktor projekt
- `text` ..... text práce
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
  - `BP_Miroslav_Spak_2017.pdf` ..... text práce ve formátu PDF