Vysoká škola ekonomická v Praze
Fakulta informatiky a statistiky
Katedra informačních technologií

Studijní program: Aplikovaná informatika
Obor: Informační systémy a technologie

Datový sklad na technologiích IBM a jeho možnosti

DIPLOMOVÁ PRÁCE

Student : Bc. Jiří Snítil
Vedoucí : doc. Ing. Jan Pour, CSc.
Oponent : doc. Ing. Ota Novotný, Ph.D.

2017

Prohlášení:

Prohlašuji, že jsem diplomovou práci zpracoval samostatně a že jsem uvedl všechny použité prameny a literaturu, ze které jsem čerpal.

V Praze dne 22. března 2017 . . . . . .

Jméno a příjmení

Poděkování

Rád bych poděkoval vedoucímu mé diplomové práce panu doc. Ing. Janu Pourouvi, CSc. za jeho odborné vedení a cenné připomínky. Také bych chtěl poděkovat společnosti IBM, která mně umožnila použít jejich software pro potřeby této práce. V neposlední řadě bych chtěl také poděkovat rodině za podporu během celého mého studia.

Abstrakt

Tato diplomová práce se zabývá analýzou rozšiřujících konceptů použitelných v datových skladech. V práci jsou vybrány tři rozšiřující koncepty k analýze a je zdůvodněn jejich výběr. Prvním z nich je způsob zachycení změn ve zdrojových systémech Change Data Capture (CDC). Druhým z nich je historizace takto zachycených změn do historické kolekce dat. Třetím z nich je použití analytických funkcí přímo v technologii datového skladu. Pro analýzu těchto vybraných rozšiřujících konceptů je vytvořeno nové testovací prostředí, v kterém je jako hlavní databázový systém použita technologie Netezza dostupná v produktu *IBM PureData System for Analytics, powered by Netezza technology* (PDA). Všechny vybrané rozšiřující koncepty jsou v tomto testovacím prostředí vyzkoušeny. Na základě výsledků z testovacího prostředí a poznatků z praxe jsou analyzovány dopady použití těchto rozšiřujících konceptů na datový sklad a to zejména vzhledem k možným přínosům.

V testovacím prostředí bylo také ověřeno, že všechny analyzované rozšiřující koncepty je možné použít v rámci datového skladu. V prvním rozšiřujícím konceptu bylo mapování LiveAudit vybráno jako vhodné pro použití při dalším zpracování dat, kdy s jeho pomocí je možné jednoznačně určit stav dat zdrojového systému v libovolném minulém časovém bodě. V druhém rozšiřujícím konceptu bylo vyzkoušeno, že data získaná pomocí mapování LiveAudit lze efektivně zpracovávat do historické kolekce dat. Na tomto základě bylo navrženo generické řešení zpracování dat ze zdrojových systémů. Ve třetím rozšiřujícím konceptu bylo vyzkoušeno, že lze pracovat v nativním analytickém prostředí RGui a přenést samotný výpočet k datům, umístěným v datovém skladu, bez nutnosti jejich migrace a že je možné vyvinout a používat nové analytické funkce napsané v jazyce C++ přímo v technologii datového skladu.

Klíčová slova

Datový sklad, rozšiřující koncepty datového skladu, Change Data Capture, CDC, historizace, historická kolekce dat, temporální data, analytika, analytické funkce, UDX, UDF, Uživatelsky definované funkce, R, Netezza, IBM PureData for Analytics, PDA, IBM InfoSphere Change Data Capture, IBM Netezza Analytics

Abstract

This diploma thesis deals with the analysis of advanced data warehouse concepts where three advanced data warehouse concepts are analysed and their selection is justified. The first selected advanced data warehouse concept is a method of capturing data changes from sources system Change Data Capture (CDC). The second concept is the historization of captured data into historical data collection. The third concept is the application of analytical functions directly within data warehouse technology. A new testing environment has been created to analyse these concepts where the main database system Netezza available in *IBM PureData System for Analytics, powered by Netezza technology* (PDA), is utilised. This testing environment allowed all selected advanced data warehouse concepts to be reviewed. An impact of the application of these advanced data warehouse concepts has been analysed based on results from the testing environment and practical insights, particularly regarding potential advances.

In the testing environment it was verified that all analysed advanced data warehouse concepts are applicable in a data warehouse. In the first advanced data warehouse concept was chosen LiveAudit mapping as appropriate for further data processing, when with this mapping it is possible to unambiguously determine the state of data in a source system at any point in the past. The second advanced data warehouse concept established that data acquired from LiveAudit mapping is possible to effectively process into historical data collection. Based on these findings, there was proposed generic solution of processing data from source systems. In the third advanced data warehouse concept was also proved, that it is possible to work in native analytic environment RGui and move the computation itself into data, which is located in the data warehouse, without the necessity of migration of these data. Further, it is possible to develop and use a new analytic function written in C++ language directly into the technology of the data warehouse.

Keywords

Data warehouse, advanced data warehouse concepts, Change Data Capture, CDC, historization, historical data collection, temporal data, analytics, analytics functions, UDX, UDF, User-defined function, R, Netezza, IBM PureData for Analytics, PDA, IBM InfoSphere Change Data Capture, IBM Netezza Analytics

Obsah

1. Úvod

Získání užitečných a přínosných informací z dostupných dat je již dlouhou dobu pro podniky nezbytné ve většině odvětví. Lze sledovat trend, že čím dál více podniků se věnuje analýze dat a pokouší se z nich získat větší množství informací, vnímá je jako zdroj, který jim může přinést konkurenční výhodu. Jedním ze způsobů jak lze dostupná data efektivně, dlouhodobě a řízeně zpracovávat je vytvořit pro ně datový sklad. Datový sklad poté slouží nejen jako uložiště daných dat, ale především umožňuje konsolidovanou práci s těmito daty, která většinou pochází z různých zdrojů, poskytuje uživatelům důvěru v tato data a zjednodušuje jejich případné další zpracování pro různorodé účely. Datové sklady byly dříve především stavěné velkými podniky například finančními institucemi, a to zejména z toho důvodu, že jejich počáteční náklady a poté i jejich provoz byl velmi finančně náročný. Datové sklady se neustále vyvíjí. Velké datové sklady se rozšiřují, aby obsáhly nové datové zdroje a různorodější zdrojová data a v poslední době se s datovými sklady lze setkat i u menších podniků, kterým pokrok v technologických řešeních umožnil jejich pořízení při přijatelných nákladech. Je možné také sledovat růst objemu dat, který je zpracováván. To je dáno například tím, že uživatelé vyprodukují těchto dat více, rozšiřuje se mezipodniková komunikace mezi informačními systémy, lze sledovat aktivitu zákazníků na internetu a i čím dál více zařízení v podnicích i doma produkuje větší objem dat, který je možné zpracovávat a získávat z nich tak přidanou hodnotu. Tím ale také vzrůstají nároky na datové sklady, které mají tyto data zpracovávat. Ať se již jedná o stávající datové sklady nebo o nově budované, je od nich požadováno, aby se přizpůsobily těmto novým požadavkům a umožnily podnikům efektivní, časově a cenově přijatelné zpracování dat a tím i získání co největší možné přidané informační hodnoty pro daný podnik při co nejmenších nákladech na vývoj a provoz.

1.1 Cíle práce

Cílem této diplomové práce je analyzovat rozšiřující koncepty použitelné v datovém skladu, vyzkoušet je v testovacím prostředí a s využitím poznatků z praxe popsat možné přínosy, nevýhody, rizika a možnosti uplatnění těchto konceptů v prostředí datového skladu. Pro splnění tohoto hlavního cíle jsou dále definovány následující tři dílčí cíle.

1) Definovat a zdůvodnit výběr konkrétních rozšiřujících konceptů datových skladů zvolených pro vyzkoušení v testovacím prostředí.

2) Vytvořit testovací prostředí a vyzkoušet použití rozšiřujících konceptů na konkrétních technologiích společnosti IBM.

3) Analyzovat dopady použití rozšiřujících konceptů na datový sklad zejména vzhledem k možným přínosům při jejich použití, případně analyzovat alternativní řešení.

Tato diplomová práce se zabývá rozšiřujícími koncepty použitelnými v datových skladech od jejich teoretického uchopení až k jejich implementaci na konkrétních technologiích v nově vytvořeném testovacím prostředí a jejich analýze. Vzhledem k širšímu rozsahu tématu je v této práci hlavní důraz kladen na výsledné vytvoření fungujícího testovacího prostředí a vyzkoušení na něm použití všech zvolených rozšiřujících konceptů.

1.2 Cílová skupina

Tato práce je určena odborné i široké veřejnosti, která se zabývá datovými sklady a jejich možnostmi. Zejména se pak jedná o odborníky, kteří se věnují datovým skladům a společnostem, které plánují vytvořit nebo rozšířit svůj datový sklad. Tyto společnosti v této práci mohou nalézt popis rozšiřujících konceptů, které rozšiřují možnosti datových skladů. Tím, že tyto koncepty jsou v této práci přímo vyzkoušené v testovacím prostředí, společnosti mohou výsledky této práce použít jako jedno z možných východisek při zvažování zavedení těchto rozšiřujících konceptů v rámci jejich stávajících nebo nově budovaných datových skladech.

1.3 Použité metody

U každé kapitoly věnující se danému rozšiřujícímu konceptu je popsáno co je cílem daného konceptu, respektive co by mělo být jeho přínosem v rámci datového skladu. Část těchto kapitol se vždy věnuje nasazení daného konceptu do testovacího prostředí a poté i jeho vyzkoušení. U každého konceptu je uveden postup, který byl zvolen pro jeho

vyzkoušení, a následně jsou popsány výsledky z testovacího prostředí a zjištění, které vyplynuly z daného testování. Každá kapitola věnující se danému konceptu také obsahuje zhodnocení konceptu, kde jsou popsány přínosy, nevýhody, rizika a možnosti uplatnění těchto konceptů při jejich použití v datovém skladu, které vychází z výsledků a zjištění, které byly získány během vyzkoušení daného konceptu v testovacím prostředí a jsou případně obohaceny zkušenostmi autora z reálných projektů, které se věnovaly daným rozšiřujícím konceptům, nebo jejich částem.

1.4 Struktura práce

Diplomová práce je rozdělena do několika celků, které na sebe navazují. Druhá kapitola je věnována komentované rešerši informačních zdrojů. V třetí kapitole se nachází teorie datových skladů a základy architektury datových skladů, které jsou potřebné k uchopení rozšiřujících konceptů, a dále je zde definováno vymezení rozšiřujících konceptů datových skladů tak, jak jsou použity dále v této práci.

Čtvrtá kapitola zdůvodňuje výběr konkrétních rozšiřujících konceptů. Zvolené rozšiřující koncepty jsou zde krátce popsány a to zejména z pohledu jejich možných přínosů při aplikaci v datovém skladu.

Pátá kapitola popisuje vytvořené testovací prostředí pro vyzkoušení rozšiřujících konceptů a věnuje se použitým technologiím. Klíčové technologie použité pro vyzkoušení rozšiřujících konceptů jsou od společnosti IBM. V závislosti na jejich použití v dalších částech práce jsou tyto technologie v různém detailu popsány včetně jejich funkcionality, použité verze a případné globální konfigurace. I když rozšiřující koncepty by měly být technologicky nezávislé (za předpokladu, že zvolená technologie poskytuje potřebná rozhraní a funkcionalitu), v dalších částech této práce je jejich implementace vyzkoušena na konkrétních technologiích. Proto i popis těchto technologií v této kapitole se zejména soustředí na části a funkcionalitu, která poté souvisí se zvolenými rozšiřujícími koncepty. Nejedná se tedy o celkový souhrn funkcionality a možností, které daná technologie obsahuje a ani se nejedná o dokumentaci produktu či jeho části.

Následují kapitoly šest až osm, které se vždy konkrétněji věnují jednomu rozšiřujícímu konceptu datového skladu. Kapitola šestá se věnuje možnostem zachytávání změn ve

zdrojových systémech. Kapitola sedmá se zabývá historizací dat a to zejména historizací po zachycení pouze změn provedených ve zdrojových systémech. Tato kapitola tak logicky navazuje na předchozí šestou kapitolu. Osmá kapitola se věnuje analytickým funkcím a jejich použití v datovém skladu. Struktura těchto všech kapitol je podobná. V jejich úvodu je daný koncept popsán včetně jeho možností a výhod, které by měl při použití v datovém skladu přinést a případně je zde i popsána další teorie související s daným konceptem či jsou zde zmíněna možná alternativní řešení. Poté je provedena implementace daného konceptu na konkrétních technologiích a celý rozšiřující koncept je vyzkoušen v testovacím prostředí. Na základě výsledků z testovacího prostředí, získaných souvisejících teoretických informací a poznatků z praxe jsou poté formulovány možné přínosy, nevýhody, rizika a možnosti uplatnění těchto konceptů v prostředí datového skladu a celý rozšiřující koncept je zhodnocen.

Poslední devátou kapitolou je závěr, který zhodnocuje splnění stanovených cílů. Tato kapitola také shrnuje podstatné výsledky, kterých bylo dosaženo v této diplomové práci.

1.5 Důvody výběru tématu

V praxi jsem se setkal s různými řešeními datových skladů. Zaujalo mě, že úzká a řekněme problémová místa se nacházela většinou na podobných místech v daných řešeních a to i přesto, že byla implementována rozdílně. Také mě zaujalo, že v některých případech byla volena záměrně méně dokonalá a funkcionalitou chudší řešení a to zejména z důvodů nižší pracnosti a tedy při stejných kapacitách s daleko rychlejší dobou implementace. Subjektivně mi přišlo, že alternativní a možná i dokonalejší řešení nebyla dostatečně zvážena a ani vyzkoušena například jen pomocí testovacích scénářů v testovacích prostředích. Pravděpodobně totiž v reálném prostředí datového skladu a konkrétních situacích nebyl dostatek času a prostředků na jejich vyzkoušení. Také zdůvodnění těchto testů a prosazení jejich financování může být problematické, zvláště pak, když se jiné, jednodušší řešení přímo nabízí k realizaci. Náročnost alternativních řešení na vývoj a jejich přínosy tak mohou být pouze odhadnuté na základě dostupných informací bez reálných podkladů ze specifického prostředí dané společnosti a může tak nastat situace, kdy není vybráno nejlepší možné řešení pro konkrétní případ. Právě z těchto důvodů jsem si vybral toto téma práce. Výstupy z této práce by měly pomoci při rozhodování, který koncept kdy

použít a zda vůbec uvažovat o jeho použití a to i proto, že dojde k jejich praktickému vyzkoušení v testovacím prostředí. Samozřejmě výsledky v testovacím prostředí nemohou nahradit test v konkrétním a specifickém prostředí reálného datového skladu konkrétní společnosti, mohou ale poukázat na přínosy, nevýhody, rizika a možnosti uplatnění těchto konceptů a poskytnout tak důležitý podklad pro zvážení použití těchto konceptů a například i podklad pro ono zdůvodnění a prosazení testů a financování těchto testů v konkrétních případech a v konkrétních datových skladech.

1.6 Předpoklady a omezení práce

Tato práce bude rozšiřující koncepty analyzovat v testovacím prostředí, v kterém budou dané koncepty implementovány na konkrétních technologiích od společnosti IBM. I s využitím teoretických znalostí a poznatků z praxe nebude možné považovat všechny zjištěné závěry za obecně platné při použití rozšiřujících konceptů s využitím jiných technologií. Při použití rozšiřujících konceptů popsaných v této práci na jiných technologiích, než s kterými pracuje tato práce (nebo i při jiných verzích těchto technologií), bude nutné vždy posoudit rozdíly mezi zvolenými technologiemi a technologiemi použitými v této práci a na základě těchto rozdílů vyvodit možné rozdíly v závěrech, které jsou popsány v této práci. Také vzhledem k neustálému zlepšování technologií a jejich lepší vzájemné integraci, je možné, že s postupem času budou také některé nevýhody a rizika zjištěné v této práci eliminovány. Odhadnutí budoucího vývoje nebo zjištění rozdílů v různých technologiích není předmětem této práce. Naopak přínosy, nevýhody a rizika rozšiřujících konceptů zjištěné při implementaci v testovacím prostředí na konkrétních technologiích mohou být unikátní a bez vyzkoušení v testovacím prostředí by nemusely být zjištěny. Mohou tak posloužit jako užitečné podněty, čemu je vhodné věnovat pozornost v případě implementace na jiných technologiích než těch, s kterými pracuje tato diplomová práce.

2. Komentovaná rešerše informačních zdrojů

Tato kapitola obsahuje komentovanou rešerši informačních zdrojů, které souvisí s tématem této diplomové práce. Informační zdroje, uvedené v této kapitole, lze rozdělit do dvou skupin. První z nich jsou infomační zdroje věnující se konkrétním technologiím použitým v rámci nově vytvořeného testovacího prostředí. Jedná se zejména o technické dokumentace a další informační zdroje vydávané společností IBM. Druhou skupinu informačních zdrojů tvoří odborné články, závěrečné práce, a příspěvky z odborných konferencí, které se věnují podobnému tématu. Pro vyhledání informačních zdrojů z druhé skupiny byly použity nástroje Summon, který agreguje informační zdroje dostupné na Vysoké škole ekonomické v Praze, Google Scholar, který se specializuje na vyhledávání textů z odborné literatury. Posledním použitým nástrojem byl systém Theses.cz provozovaný Masarykovou universitou, který umožňuje vyhledávat v závěrečných pracích.

Mezi použité informační zdroje spadající do první skupiny patří zmíněná dokumentace k jednotlivým použitým technologiím. Jedná se zejména o online dokumentaci (IBM, 2016a) k databázovému systému IBM® PureData™ System for Analytics, powered by Netezza technology ve verzi 7.2.1. Tento systém tvořil hlavní část testovacího prostředí, kdy tato databáze byla v roli datového skladu. Dalším použitým informačním zdrojem byla online dokumentace (IBM, 2015a) k produktu InfoSphere Change Data Capture ve verzi 11.3.3. Tento produkt byl použit v testovacím prostředí v rámci vyzkoušení prvního rozšiřujícího konceptu. Použitou online dokumentací také byla dokumentace (IBM, 2017a) k databázovému systému DB2 for LUW ve verzi 11.1, která byla použita v roli zdrojových systémů.

Všechny tyto uvedené informační zdroje byly intenzivně využívány, zejména pro zprovoznění technologií v rámci testovacího prostředí, případně jejich vhodné nastavení tak, aby je bylo možné použít pro vyzkoušení zvolených rozšiřujících konceptů v této práci. V případě, že byly použity další informační zdroje, které spadají do této první skupiny, jsou vždy tyto zdroje uvedeny přímo v rámci jednotlivých kapitol, kde jsou použity.

Při provádění rešerše pro druhou skupinu informačních zdrojů bylo nalezeno velké množství prací českých i zahraničních autorů, které se zabývají datovými sklady. Tyto práce se často věnovaly tvorbě konkrétního řešení datového skladu například

(Plaček, 2016), (Šťava, 2015), (Havlas, 2015) nebo konkrétním technologiím použitelným pro datové sklady například (Kuželka, 2015), (Šiler, 2012). Nepodařilo se ale nalézt žádnou práci, která by se uceleně věnovala zvoleným rozšiřujícím konceptům datových skladů, kterým se věnuje tato práce, a jejich implementaci či vyzkoušení v konkrétních technologiích společnosti IBM.

V rámci rešerše se podařilo nalézt práce, které se zvolenými rozšiřujícími koncepty souvisí, například zmiňují různé principy použité v rámci rozšiřujících konceptů nebo zmiňují či popisují použité technologie.

Pro první zvolený rozšiřující koncept, který se zabývá možnostmi zachytávání změn ve zdrojových systémech, lze zmínit práce (Hejmalíček, 2015) a (Dvorský, 2016), které zmiňují tento způsob získávání zdrojových dat pomocí CDC (Change Data Capture, tedy zachycení změn) jako alternativu k ETL. Tyto práce se však CDC nezabývají a ani ho nepoužívají. Za zmínku stojí také příspěvek ve sborníku (Tank, et al., 2010) z mezinárodní konference ARTCom, který se zabývá možnostmi zrychlení ETL nad daty získanými pomocí CDC.

Druhý zvolený rozšiřující koncept se zabývá historizací dat v případech, kdy jsou zachyceny pouze změny, které byly provedeny ve zdrojových systémech. Tedy tomu, jak lze vytvářet historické kolekce dat bez nutnosti mít k dispozici celý snímek zdrojových dat, kdy v rámci kapitoly věnující se tomuto konceptu je v této práci provedena i jeho implementace. Nepodařilo se nalézt žádné práce, které by se detailněji věnovaly tomuto tématu, či se zabývaly konkrétním řešením jeho implementace.

Byly nalezeny práce, které historizaci v datovém skladu zmiňují, například ji v určité části řeší pomocí specifické komponenty nástroje ETL (Rokyta, 2013) či popisují různé varianty řešení historizace dat v datovém skladu z konkrétního projektu na obecnější úrovni (Hník, 2013). V této práci je také uvedeno, že konkrétní řešení historizace dat bylo řešeno pomocí specializované komponenty zakoupené od externího dodavatele, popis fungování této komponenty ale v práci není uveden. Historizace dat také není hlavním tématem těchto nalezených prací.

Lze nalézt publikace, které se obsáhle věnují problému temporálních dat, které s historizací souvisí, například publikace (Johnston a Weis, 2010), kde lze nalézt detailní možnosti práce s těmito daty. S tímto tématem také nelze opomenout publikaci Ralpha Kimballa

(Kimball et. al., 2013), který se této problematice věnuje v souvislosti s pomalu měnícími se dimenzemi (Slowly Changing Dimensions, zkráceně také SCDs). Také lze nalézt odborné články, které se věnují specifickým algoritmům, které lze nad daty historické kolekce použít například (Zhou et al 2006).

Poslední třetí rozšiřující koncept se věnuje analytickým funkcím a jejich použitím v datovém skladu. Tato část práce se detailněji věnuje použití programovacího jazyka R a C++ přímo v technologii datového skladu. Existuje obrovské množství závěrečných prací, které se věnují těmto jazykům nebo analytickým funkcím a možnostem datové analýzy. Nepodařilo se ale nalézt závěrečné práce věnující se analytickým funkcím použitým přímo v platformě datového skladu, respektive dopadům, které toto použití může mít pro datový sklad a datové analytiky na základě vyzkoušení tohoto použití v testovacím prostředí.

Při rešerši týkající se podobného tématu byl nalezen zajímavý článek (Lopez a D'Antoni, 2014) v odborném časopise, který se věnuje architektuře datového skladu pro analytické použití nad velkými objemy dat. Tento článek se zabývá přidáním platformy Hadoop do architektury datového skladu a právě použitím této platformy pro specifické analytické úlohy.

3. Datový sklad a vymezení rozšiřujících konceptů

Tato kapitola si klade za cíl popsat základní teorii datových skladů a základy architektury datových skladů a na tomto základě poté definovat vymezení rozšiřujících konceptů datových skladů pro použití v této práci.

3.1 Vymezení datového skladu

Samotná definice datového skladu je poměrně problematická a lze se setkat s různými definicemi od různých autorů. Poměrně známá a rozšířená definice je od Williama Inmona, který definuje datový sklad jako subjektově orientovanou, integrovanou, změnám nepodléhající, historickou kolekci dat používaných na podporu rozhodování (Inmon a Linstedt, 2015). Subjektivně orientovanou kolekcí je zde myšleno, že datový sklad se soustředí na danou oblast či oblasti, kterým například může být zákazník, objednávka, produkt, zatímco tradiční transakční systémy se většinou orientují na funkční oblasti, jakými jsou například zpracování pojištění, provedení obchodních transakcí a podobně. Druhou charakteristikou je, že datový sklad je integrovaný, to znamená, že data jsou vhodně transformována z různých zdrojových systémů a nahrána do datového skladu v takové podobě, aby tvořila jednotnou datovou základnu. Třetí charakteristikou je změnám nepodléhající, to znamená, že na rozdíl od transakčních systémů, kde mohou být data modifikována, v datovém skladu se data po nahrání již nemodifikují. Čtvrtou a poslední charakteristikou je, že kolekce dat je historická. To znamená, že u všech dat v datovém skladu lze říci, zda jsou nebo kdy byly platná (Inmon, 2005).

Druhou také poměrně známou a rozšířenou definicí je definice od Ralpha Kimbala, který definuje datový sklad jako systém, který extrahuje, čistí, přizpůsobuje a dodává zdrojová data do dimenzionálního datového uložiště a poté podporuje a implementuje dotazování a analýzy pro účely rozhodování (Kimbal, 2004).

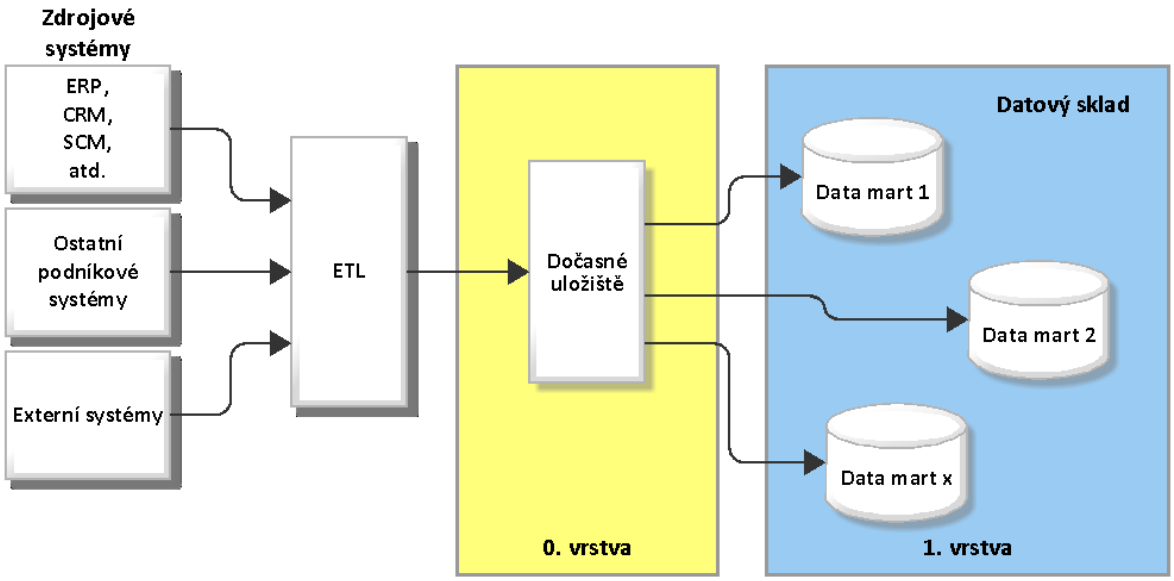
William Inmon i Ralph Kimbal jsou uznávanými odborníky na datové sklady a jejich tvorbu. Jejich definice datových skladu jsou poměrně rozdílné. Například v druhé definici

je uvedeno, že součástí datového skladu je i extrakce zdrojových dat, což z pohledu první definice do datového skladu nespadá. Pro tuto diplomovou práci je zejména důležité vymezení šíře tohoto pojmu. Vzhledem k tomu, že některé rozšiřující koncepty, kterými se v dalších kapitolách zabývá tato práce, je možné uplatnit nejen na specifickou kolekci dat, tak jak datový sklad definuje William Inmon, ale i na okolí takto vnímaného datového skladu, bude tato práce pracovat s širším vymezením tohoto pojmu.

3.2 Architektura datového skladu

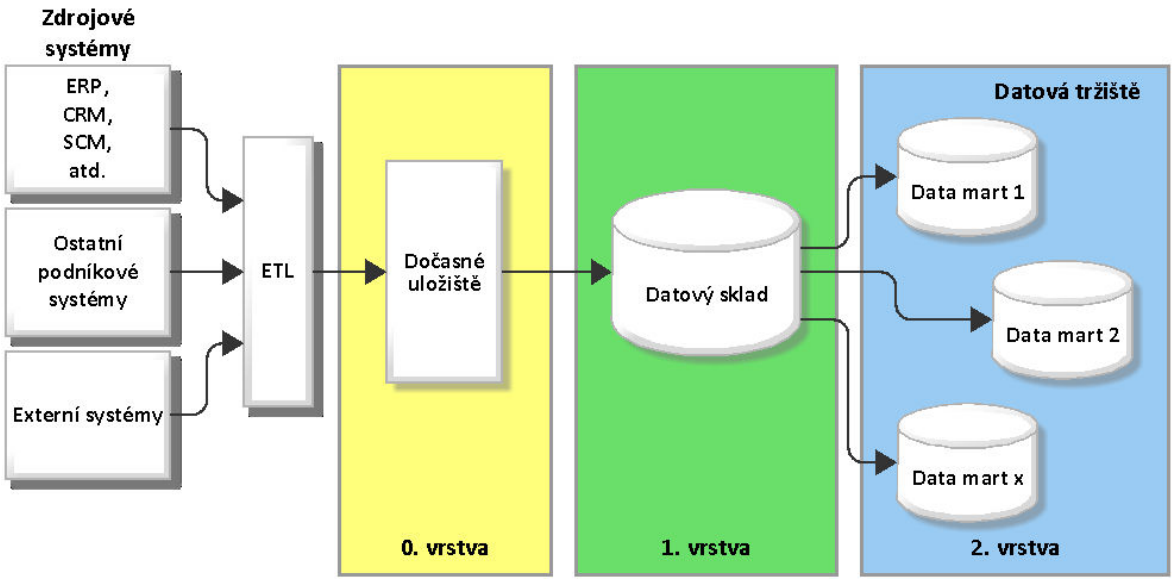
Přístupů k architektuře datových skladů existuje také více. Dvě základní a asi nejznámější architektury jsou Architektura nezávislých datových tržišť a Architektura konsolidovaného datového skladu.

Architektura nezávislých datových tržišť vychází z principu vytváření relativně nezávislých datových tržišť pro specifické útvary podniku. Datový sklad je poté tvořen množinou těchto tržišť. Tyto datová tržiště obvykle obsahují vrstvy a komponenty tak, aby mohly nezávisle existovat na ostatních. Návrh této architektury byl popsán Ralphem Kimballem. Výhodou tohoto přístupu je především relativní rychlost vybudování tohoto řešení. Mezi hlavní nevýhody patří následná integrace jednotlivých datových tržišť a například i zajištění jednotné reportingové vrstvy a s tím spojené potencionální riziko rozdílné interpretace skutečnosti (Novotný, et al., 2005). Pro přehlednost je tato architektura zobrazena na následujícím obrázku (Obr. 1).



Obr. 1: Architektura nezávislých datových tržišť zpracováno dle (Pour, et al., 2012)

Druhou zmíněnou architekturou je Architektura konsolidovaného datového skladu. Oproti předchozí architektuře zde přibývá vrstva mezi dočasným uložištěm a datovými tržišti. Tato vrstva obsahuje data, která by měly pokrývat všechny definované analytické potřeby organizace. Z této vrstvy poté vznikají jednotlivá datová tržiště. Tato architektura byla popsána Williamem Inmonem. Mezi hlavní výhody tohoto řešení patří flexibilita, kdy lze poměrně jednoduše vytvářet další datová tržiště nebo jiné struktury pro specifické použití dat z již předpřipravené datové základny. V tomto řešení se značně eliminuje riziko rozdílné interpretace skutečnosti a také odpadá duplikace komponent řešení, tak jak mohou nastat u architektury nezávislých datových tržišť (Novotný, et al., 2005). Mezi hlavní nevýhody lze zařadit větší finanční náročnost celého řešení, delší doba implementace a s tím související rizika spojená se změnou zdrojových systémů nebo business požadavků (Pour, et al., 2012). Tato architektura je zobracena na následujícím obrázku (Obr. 2).



Obr. 2: Architektura konsolidovaného datového skladu zpracováno dle (Pour, et al., 2012)

Na obrázcích jsou patrné i další komponenty používané v architektuře datového skladu jako jsou například datová tržiště (data marts), dočasné uložiště (data staging areas) a ETL (transformační nástroje v češtině často označované jako datové pumpy). Tyto komponenty mohou být považovány za jednotlivé nástroje Business Intelligence (Novotný, et al., 2005), jak je ale z obrázků patrné lze je v širším pojetí také vnímat jako vrstvy architektury datového skladu. V již uvedené publikaci (Novotný, et al., 2005), je také uvedeno, že existuje na tuto problematiku uspořádání komponent respektive nástrojů více názorů a lze se setkat s různou kategorizací. Ostatně i Ralph Kimball v publikaci (Kimball, 2013) v kapitole první zahrnuje v rámci architektury i ETL nástroje a celou tuto architekturu označuje jako DW/BI (Data Warehouse and Business Intelligence). Tento spojitý pojem je potom používán v celé publikaci, aniž by jeho části byly přesněji vymezeny. I z toho lze usoudit, že hranice a překryv těchto dvou pojmů je u různých autorů odlišný a jejich obecné vymezení je problematické. Jak již bylo zmíněno, vzhledem k tomu, že některé rozšiřující koncepty je možné uplatnit na různých vrstvách v architektuře datového skladu, respektive na různé komponenty, bude tato práce pracovat s širším vymezením datového skladu tak, aby bylo možné jednotlivé komponenty považovat i za vrstvy řešení datového skladu. Protože tyto zmíněné komponenty úzce souvisí s touto prací, jsou krátce popsány v následujících podkapitolách. Tyto podkapitoly čerpají z (Kimball, 2014), (Inmon a Linstedt, 2015), (Novotný, et al., 2005) a (Pour, et al., 2012).

3.2.1 ETL – Transformační nástroje

ETL je zkratkou pro Extraction, Transformation a Load, v češtině jsou často tyto nástroje označované jako datové pumpy. Hlavním úkolem ETL je přenos dat z jednoho či více systémů (často zdrojových systémů), transformace dat do nové vhodné podoby a nahrání těchto dat do cílového systému nebo systémů. Tyto nástroje mohou sloužit i jen jako zprostředkovatelé přenosu dat mezi systémy. Mohou ale řešit i složité transformační úlohy, jako je například překlad různých datových typů, sjednocení významu atributů, kdy stejná skutečnost je v různých systémech uložena odlišně (nebo obráceně je uložená stejně, ale vyjadřuje jinou skutečnost například z důvodů odlišné terminologie), a také mohou definovat pravidla týkající se kvality dat.

3.2.2 DSA – Dočasné uložiště

Dočasné uložiště dat zkratkou DSA (Data Staging Area, někdy také v praxi označována pouze jako Stage) slouží k dočasnému uložení extrahovaných dat z produkčních systémů. Data v této komponentě jsou uložena ve stejné struktuře jako v produkčních systémech, nebyla tedy transformována, obohacena či jinak změněna. V praxi se často používá označení, že data jsou ve formátu jedna ku jedné, nebo že tato vrstva obsahuje snímek zdrojových systémů k určitému časovému okamžiku. Charakteristické pro data v této vrstvě je to, že v momentě, kdy jsou data dále zpracována, jsou zároveň odstraněna z této vrstvy.

3.2.3 DMA – Datová tržiště

Datová tržiště označovaná také jako Data Mart (DMA) obvykle obsahují data pro pokrytí určité problematiky, často konkrétního okruhu uživatelů (oddělení, divize, pobočka apod.). Jak je patrné z kapitoly 3.2 Architektura datového skladu, existují dva základní způsoby uchopení datových tržišť. První z nich je zobrazen na obrázku číslo jedna, kdy datová tržiště jsou na sobě nezávislá a jejich množina tvoří datovou základnu a datový sklad. Druhý způsob je zobrazen na obrázku číslo dva, kdy datová tržiště vznikají z jednotné datové základny. V obou případech však platí, že jednotlivá datová tržiště jsou specializovaná na určitou potřebu uživatelů a jednotlivě neobsahují všechna data.

3.2.4 DWH – Datový sklad

Datový sklad (Data Warehouse) zkratkou označovaný také jako DWH nebo DW je již popsán v kapitole 3.1 Vymezení datového skladu a v kapitole 3.2 Architektura datového skladu. V této kapitole je uvedeno a popsáno jeho užší i širší chápání. Tato práce k tomuto pojmu přistupuje v jeho širším pojetí. Kromě tohoto pojetí datového skladu se lze setkat i s jeho užším významem, kdy je datový sklad považován jen jako komponenta řešení architektury DW/BI. V praxi se také tento užší význam označuje jako jádro datového skladu a je znázorněno jako první vrstva na obrázku číslo dvě. Při velkých implementacích datových skladů je běžné, že i toto jádro se skládá z více vrstev. Jak již bylo popsáno, tato práce přistupuje k pojmu datového skladu v jeho širším pojetí, pokud bude potřeba ho použít v jeho užším pojetí, bude vždy explicitně uvedeno, že se jedná o jádro datového skladu či o komponentu architektury datového skladu.

3.2.5 ODS – Operativní uložiště dat

Operativní uložiště dat (Operational Data Store, zkráceně ODS) je další komponentou používanou v architektuře datových skladů. Jedná se o uložiště dat, které uživatelům zpřístupňuje aktuální a integrovaná data ze zdrojových systémů. ODS obsahuje data subjektově orientovaná a integrovaná. Data v ODS na rozdíl od užší definice datového skladu mohou podléhat změnám a nejsou historická. Použití ODS v architektuře datového skladu se může výrazně lišit. ODS může využívat data přímo ze zdrojových systémů nebo z jádra DWH (případně kombinace obou), ale také může sloužit jako další zdroj dat pro jádro DWH. Popis těchto různých typů použití ODS lze například nalézt v (Inmon a Linstedt, 2015). Inmon zde také dělí komponentu ODS na tři různé třídy, které se liší aktuálností dat ze zdrojových systémů. Komponenta ODS není znázorněna na obrázcích jedna a dva, některé rozšiřující koncepty s ní ale mohou souviset, a proto je v tomto odstavci krátce popsána.

3.3 Vymezení rozšiřujících konceptů datových skladů

Nepodařilo se nalézt ustálenou a obecně přijímanou definici rozšiřujících konceptů datových skladů, která by se v literatuře běžně používala. Toto spojení je proto v této práci

chápáno ve svém obecném smyslu, tedy že rozšiřující koncept rozšiřuje možnosti datového skladu tím, že buď přidává novou funkcionalitu, nebo řeší jeho určitý generický problém. Tento rozšiřující koncept by měl být do značné míry obecný a tedy aplikovatelný na různé datové sklady za předpokladu, že použité technologie datového skladu obsahují potřebnou funkcionalitu, případně rozhraní, které daný rozšiřující koncept potřebuje. Každý koncept může mít také různé předpoklady a omezení použití, které je nutné před použitím daného konceptu zvážit. Pokud však jsou předpoklady splněné a zvolený koncept je vhodný k použití v daném datovém skladu může být tato obecnost konceptu výhodná, protože je možné daný koncept aplikovat pro konkrétní případ použití obvykle jen s malými modifikacemi. Tím se mohou tyto rozšiřující koncepty stát relativně levnými pro vývoj a nasazení vůči jiným alternativám. Je však vždy nutné zvážit implementační specifika konkrétního datového skladu. Tato práce bude rozšiřující koncepty analyzovat v testovacím prostředí, v kterém budou dané koncepty implementovány na konkrétních technologiích od společnosti IBM.

4. Výběr rozšiřujících konceptů

Cílem této kapitoly je zdůvodnit výběr rozšiřujících konceptů a krátce je popsat. Při popisu rozšiřujících konceptů je kladen důraz především na přínosy, kterých může být dosaženo při použití daného konceptu v datovém skladu. Teorie související s daným konceptem a jeho detailní popis včetně jeho analýzy je uveden až v kapitole věnující se přímo danému konceptu.

4.1 Zachycení změn ve zdrojových systémech

Prvním vybraným rozšiřujícím konceptem je Change Data Capture, zkráceně CDC. Tento koncept se v architektuře datového skladu obvykle nachází mezi zdrojovými systémy (zejména jejich databázemi) a dočasným uložištěm dat. Cílem tohoto konceptu je získávat informace o změnách v datech ve zdrojových databázích s jejich co nejmenším vytížením. CDC tedy nabízí alternativu k použití ETL, které vždy zatíží zdrojovou databázi, a to z důvodu, že se do zdrojové databáze připojuje přes konektor, z kterého čte data (nehledě na to, zda se jedná o nativní konektor konkrétní databáze nebo o generický konektor, jakým je například ODBC či JDBC). Naproti tomu CDC se dokáže vyhnout tomuto zatížení, protože dokáže změny provedené v datech vyčítat z databázových logů, obvykle se tedy nevytváří žádné databázové připojení, databáze nemusí tvořit exekuční plán a provádět načítání dat z disků. Jedním z nástrojů, které je možné použít pro implementaci tohoto rozšiřujícího konceptu je například IBM InfoSphere Change Data Capture, který je použit s dvěma databázemi v testovacím prostředí pro vyzkoušení tohoto konceptu. Hlavním přínosem implementace tohoto konceptu v testovacím prostředí by mělo být zjištění dopadů na datový sklad a jeho vrstvy při použití této komponenty a odhalení případných rizik spojených s tímto rozšiřujícím konceptem. Tento rozšiřující koncept je vybrán především z důvodů velkých potencionálních přínosů na straně zdrojových systémů a také z důvodu prověření dopadů na datový sklad a jeho vrstvy.

4.2 Historizace

Druhým vybraným rozšiřujícím konceptem je historizace. Tento koncept se zabývá splněním části definice datového skladu od Williama Inmona popsané v kapitole 3.1 Vymezení definice datového skladu, konkrétně tomu, že datová kolekce je historická. To znamená, že je možné určit, kdy která data byla či jsou platná. Z jiného úhlu pohledu lze také říci, že všechna data v datovém skladu určitým způsobem mají dimenzi času. Jelikož se jedná o základní funkčnost datového skladu (ať v užším či širším pojetí) existuje mnoho řešení jak splnit tuto definiční podmínku. Tento koncept v rámci této práce si neklade za cíl, zmapovat, implementovat nebo nalézt optimální řešení splnění definiční podmínky. Tento rozšiřující koncept se v této práci zabývá možností historizace po použití konceptu CDC, tedy tím, jaký dopad bude mít pro datový sklad situace, kdy přicházejí ze zdrojových systémů pouze provedené změny. Tím tedy tento koncept logicky navazuje na předchozí rozšiřující koncept. Z těchto důvodů byl daný rozšiřující koncept zvolen k analýze a vyzkoušení v testovacím prostředí.

4.3 Analytické funkce a datový sklad

Třetí vybraný rozšiřující koncept se zabývá analytickými funkcemi a souvisejícími možnostmi datového skladu. Analytických nástrojů, které využívají data z datových skladů, je celá řada. Tyto nástroje svým uživatelům poskytují rozsáhlé možnosti analýzy dat. V praxi se lze setkat s častým použitím těchto nástrojů ve formě tlustých klientů případně dedikovaných serverů. Toto použití je vzhledem k vývoji a způsobu využívání těchto analytických nástrojů logické. V začátcích při malých datových množinách je možné tyto nástroje použít přímo na klientských stanicích, kdy data jsou prvně načtena z datového zdroje (často z datového skladu) do tohoto nástroje a poté jsou v tomto nástroji analyzována.

S rostoucí množinou dat začíná být toto použití problematické, a to především z důvodů nedostatečných výpočetních zdrojů na straně klientské stanice. Jedním z nabízejících se řešení je vytvoření dedikovaného analytického serveru. Tento server je daleko lépe škálovatelný a náročnost výpočtu je na tento server přenesena z klientské stanice. S dále rostoucí množinou dat nutnou k analýzám se objevují problémy i při použití dedikovaného

analytického serveru. Jedná se například o problémy s migrací dat. Pokud množství dat začne přesahovat velikost stovek gigabitů, stává se migrace obtížná. Typicky se jedná o zatížení přenosové infrastruktury, zdrojového systému i cílového systému. Toto zatížení nemusí být kritické, obvykle ale dochází ke značným zpožděním. V běžné praxi také nelze očekávat, že všechny tři zmíněné prvky budou plně vyhrazené analytickým potřebám. Tyto prvky musí plnit svoji obvyklou funkcionalitu a nelze je tedy dedikovat pro migraci dat. To vede buď k ještě větším prodlevám, nebo k přesunutí migrací dat do časového okna, kdy tyto systémy jsou méně využívané. V obou případech je uživatel nucen důkladněji plánovat harmonogram práce a to, jaká data budou kdy potřeba na analytickém serveru.

V případě nutnosti ad-hoc analýzy nad jinými než připravenými daty na analytickém serveru mohou být doby dodání této analýzy velmi dlouhé. Dalším možným problémem, který se objevuje při velké množině dat určené k analýzám, je samotná škálovatelnost analytického serveru. Může se jednat o problémy spojené hardwarovým škálováním, cenou za dodatečné licence, které pokryjí výkonové nároky nebo i o samotný software, který musí umožnit daný hardware efektivně využít, například tím, že je schopný danou úlohu vhodně paralelizovat. Dalšími nevýhodami může být samotné uložení takto velké množiny dat na analytickém serveru, kdy v podstatě dochází k jejich duplikaci.

Jedním z možných řešení těchto problémů může být přenesení výpočtu přímo do datového skladu, kdy je uživateli nabídnuta podobná funkcionalita a prostředí jako v případě analytických aplikací. Toto použití datového skladu klade nároky na použitou technologii datového skladu, která musí toto použití vhodně podporovat tak, aby samotný výpočet mohl probíhat přímo v datovém skladu. Výhodou tohoto použití datového skladu je, že nedochází k nutnosti migrovat data a tedy není nutné vytvářet celá řešení kolem této migrace. Veškerá data uložená v datovém skladu také mohou být okamžitě dostupná pro analytické potřeby. V případě hlubších analýz a například při vytváření různých modelů může analytikům poskytnout větší efektivitu práce a výrazně zkrátit dobu potřebnou pro dodání výsledků. Tento koncept je zvolený pro analýzu a vyzkoušení v testovacím prostředí především s ohledem na možné přínosy uvedené výše. Jelikož se jedná o velmi rozsáhlé téma je hlavní důraz kladený především na možnosti praktického využití tohoto rozšiřujícího konceptu a na odzkoušení daného konceptu v testovacím prostředí. Tomuto rozšiřujícímu konceptu je v této práci věnován menší rozsah v porovnání s předchozími dvěma rozšiřujícími koncepty.

5. Testovací prostředí a technologie

Tato kapitola se zabývá technologiemi, které byly použity pro vytvoření testovacího prostředí. Cílem této kapitoly je stručně popsat hlavní technologii použitou v testovacím prostředí, kterou je technologie Netezza dodávaná v rámci produktu IBM PureData for Analytics, a zmínit další technologie použité v testovacím prostředí pro vyzkoušení rozšiřujících konceptů. Těmto dalším technologiím použitým v rámci této práce se již tato kapitola detailněji nevěnuje. Tyto technologie jsou detailněji popsány u konkrétních rozšiřujících konceptů, kde je i popsáno jejich začlenění do testovacího prostředí tak, aby technologie přímo související s daným rozšiřujícím konceptem byla uvedena přímo u tohoto konceptu.

5.1 IBM Netezza

IBM Netezza je databázová technologie určená pro datové sklady a pro pokročilou datovou analytiku. Původně tato technologie byla vyvíjena společností Netezza Corporation, která byla v roce 2010 koupena společností IBM (IBM, 2010). Po akvizici byla Netezza zařazena do rodiny produktů IBM PureSystems a byla přejmenována na IBM PureData System for Analytics, powered by Netezza technology. Zkráceně se také používá název PDA nebo se lze setkat také s označením, že se jedná o produkt IBM Netezza Data Warehouse Appliance (IBM, 2017b). IBM PureData System for Analytics, powered by Netezza technology je profilována jako takzvaná Data Warehouse Appliance, tedy integrované zařízení datového skladu, které v sobě kombinuje hardware i software.

Pro určité zjednodušení je v této práci používán obecně známý název IBM Netezza nebo pouze Netezza. Tento název je používán záměrně i proto, že pro potřeby této práce je podstatnější samotná databázová technologie Netezza, než celé konkrétní zařízení datového skladu včetně dané hardwarové konfigurace (Data Warehouse Appliance).

IBM Netezza je použita ve všech rozšiřujících konceptech, kterým se věnuje tato diplomová práce. V prvním rozšiřujícím konceptu je použita jako cílová databáze, do které jsou nahrávány zachycené změny ze zdrojových systémů. V druhém rozšiřujícím konceptu je v této technologii vyvinuto řešení, které umožňuje provést zpracování zachycených

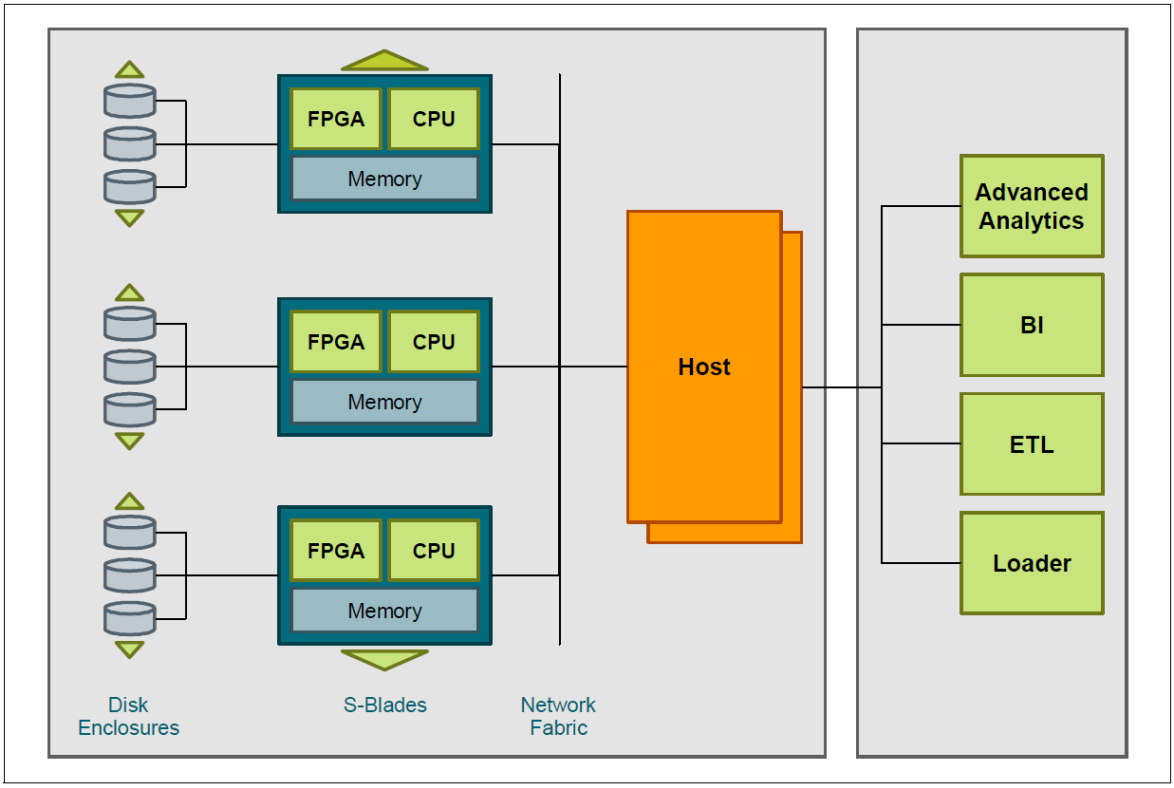
změn ze zdrojových systémů do historické kolekce dat. V rámci třetího rozšiřujícího konceptu jsou v této technologii vyzkoušeny analytické funkce a mimo jiné jsou pro tuto technologii vyvinuty nové analytické funkce pomocí jazyka C++.

IBM Netezza tak tvoří hlavní část vytvořeného testovacího prostředí. Pro vyzkoušení rozšiřujících konceptů v testovacím prostředí byla použita technologie Netezza ve verzi 7.2.1, konkrétně se jedná o tzv. Netezza Platform Software 7.2.1 (zkráceně NPS), který představuje samotný databázový systém. Pro potřeby testování byla v tomto systému zapnuta plná podpora databázových schémat.

5.1.1 Architektura technologie IBM Netezza

Technologie Netezza integruje zpracovávání dat, datové uložiště a databázi do jednotného systému, který je navržen tak, aby umožňoval flexibilní škálování. Architektura této technologie je založena na několika principech, mezi které patří přesun zpracování co nejblíže k datům, vyvážená, masivně paralelní architektura, jednoduchost, flexibilní konfigurace a extrémní škálovatelnost (Francisco, 2014).

Tato technologie nabízí velmi vysoký výkon při zpracovávání velkých objemů dat. Tohoto vysokého výkonu je z velké části dosaženo právě díky unikátní architektuře asymetrického masivně paralelního zpracování (v angličtině Asymmetric Massively Parallel Processing architecture, často zkracováno na AMPP). Tato architektura je přehledně znázorněna na následujícím obrázku číslo 3.



Obr. 3 – Architektura AMPP technologie Netezza (Francisco, 2014)

Na tomto obrázku jsou zobrazeny všechny nejdůležitější části této technologie. V levé části obrázku jsou znázorněny disková pole (Disk Enclosures), které slouží jako uložiště dat. Tyto disková pole jsou propojena přes vysokorychlostní sít se Snippet Blades (S-Blades, také označováno jako Snippet Processing Units, zkráceně SPU). Každý tento S-Blade tvoří základní výkonovou jednotku tohoto masivně paralelního systému. Jak je i znázorněno na obrázku jedná se v zásadě o samostatný server, který má k dispozici výkonný více jádrový procesor, vlastní RAM paměť a specializovaný FPGA čip (Field Programmable Gate Array), který je uzpůsobený pro výrazné zrychlení základních databázových operací (Francisco, 2014).

Standardizované rozhraní a tedy i přístup k celému systému zajišťuje tzv. Host (na obrázku podbarven oranžově), který koordinuje práci S-Blades, vytváří databázové plány a kompiluje dotazy do exekučních kódů. K tomuto Hostu se připojují uživatelé i externí aplikace (Francisco, 2014). Příklady takovýchto aplikací jsou na obrázku zobrazeny vpravo a jsou zeleně podbarveny.

5.2 Další technologie a produkty použité v rámci testovacího prostředí

Pro vyzkoušení konkrétních rozšiřujících konceptů v testovacím prostředí bylo toto prostředí rozšířeno o další technologie. Tyto technologie jsou detailněji popsány přímo v kapitolách, které se věnují daným rozšiřujícím konceptům.

Pro první zvolený rozšiřující koncept, který se zabývá zachycením změn ve zdrojových systémech, se jedná zejména o technologii CDC, která je obsažena v produktu IBM InfoSphere Change Data Capture. Tento produkt byl začleněn do testovacího prostředí včetně jeho komponent, kterými jsou Access Server, Management Console, Replication Engine for DB2 a Replication Engine for IBM Netezza. Tento produkt a jeho jednotlivé komponenty jsou podrobněji popsány v kapitole šesté, která se věnuje tomuto konceptu. Pro tento rozšiřující koncept byla také použita databáze DB2 for LUW, která představovala zdrojový systém.

Pro další rozšiřující koncept, věnující se historizaci, nebylo potřeba rozšířit testovací prostředí o další technologie. Řešení historizace bylo vyvinuto přímo v technologii IBM Netezza s využitím funkcionalit, které tato technologie obsahuje.

Pro poslední zvolený rozšiřující koncept, který se věnuje analytickým funkcím, bylo testovací prostředí rozšířeno o komponentu IBM Netezza Analytics a o klasickou distribuci jazyka R ve verzi 3.3.2. Také byla použita funkcionalita prostředí systému Netezza pro kompilaci nově vyvinutých zdrojových kódů C++.

V následujících třech kapitolách jsou postupně popsány jednotlivé zvolené koncepty včetně dopadů při jejich použití do datových skladů, výsledky zjištěné při implementaci v testovacím prostředí a právě i zde zmíněné technologie a s nimi související produkty.

6. Change Data Capture (CDC)

Tato kapitola se věnuje rozšiřujícímu konceptu Change Data Capture, tedy tomu, jak lze zachytávat změny ve zdrojových systémech s jejich co nejmenším vytížením. Také se zabývá dopady, které má tento koncept na datový sklad. Cílem této kapitoly je popsat základní principy fungování CDC a možnosti získávání dat ze zdrojových systémů, stručně popsat produkt zvolený k otestování konceptu CDC v testovacím prostředí, zejména části, které jsou použité dále v této kapitole. Dalším cílem je začlenit tento produkt do testovacího prostředí a na základě testovacího postupu vyzkoušet tento koncept v testovacím prostředí. V závěru kapitoly jsou uvedené výsledky a zjištění z testovacího prostředí a celý koncept je zhodnocen.

6.1 Základní principy fungování CDC

Principem fungování CDC je schopnost zachytávat změny v datech ve zdrojových systémech. Snahou použití CDC je získávat tyto změny efektivně, a pokud je to možné, dosáhnout co nejmenšího zatížení těchto zdrojových systémů. Získání zdrojových dat by tedy nemělo mít žádný anebo pouze zanedbatelný dopad do běžného používání zdrojových systémů a to zejména tak, aby nebyla narušena jejich primární funkce.

6.2 Možnosti získání dat ze zdrojových systémů

Zdrojové systémy mohou obsahovat velké množství dat, jsou ale obvykle stavěné pro práci s relativně menší množinou dat s velmi krátkou dobou odezvy. Často u nich manipulace s daty probíhá na úrovni jednotlivých prvků (v relačních databázích na úrovni jednotlivých záznamů), kdy pomocí různých technik (například indexace dat) je dosahováno velmi nízkých časů, které jsou potřebné k provedení operace.

Zdrojové systémy nejsou obvykle stavěné pro analytické potřeby a nejsou optimalizované pro analytické zpracovávání dat. Případné zpracování dat pro analytické potřeby přímo ve zdrojových systémech by mohlo vést k jejich nepřípustnému zatížení. V případě potřeby analytického zpracovávání dat a/nebo tvorby rozsáhlých analýz na základě těchto dat, je

skoro vždy nutností, tato data přesunout do jiného systému, který je na hromadné a analytické zpracování dat přizpůsoben. Tímto systémem může být například datový sklad.

Nastává tedy otázka, jak tato data z těchto systémů získat. Existuje několik možností, jak k tomuto problému přistoupit. Tyto možnosti jsou dále krátce popsány v následujících subkapitolách. Při zvažování, kterou možnost zvolit, je potřeba mít vždy na paměti typické fungování zdrojových systémů.

Zdrojové systémy jsou odladěné na jejich primární úkol, kterým může být například podpora běžného provozu podniku. Často platí, že čím důležitější data daný systém má, tím je kritičtější pro chod podniku. Výpadek těchto zdrojových systémů nebo zvýšená doba jejich odezvy může mít velmi negativní dopad na chod podniku a může mu to způsobit značné komplikace nebo i finanční ztráty.

Z toho vyplývají i omezení, které se vztahují na získávání dat z těchto systémů. Při práci s těmito systémy nesmí dojít k jejich většímu zatížení, které by vedlo k ohrožení jejich primární funkce jako je například růst dob odezvy nad únosnou mez.

Je také potřeba počítat s tím, že jejich zastavení nebo omezení chodu nemusí být možné a i v případě doby, kdy je jejich vytížení menší, může mít přednost jejich zálohování a pravidelná údržba. V následujících subkapitolách jsou krátce popsány možnosti získání dat ze zdrojových systémů právě vzhledem ke zmíněným omezením.

6.2.1 Využití funkcionality zdrojových systémů

Tento způsob získání zdrojových dat není příliš obvyklý. Jedná se o případy, kdy se samotná aplikace zdrojového systému stará i o zachycení změn a jejich předání do cílového systému nebo uložiště.

Reálná implementace často vypadá tak, že daná vrstva aplikace, která zajišťuje komunikaci s persistentní databázovou vrstvou, uloží změny jednak v primárním a rychlém databázovém systému aplikace a také asynchronně zašle změny i do další komponenty, která zajistí například dávkové nahrání změn do cílového systému nebo uložiště.

Tento přístup má několik výhod. Například je zachováno transakční rychlé zpracování typické pro zdrojové systémy, protože aplikace nečeká na potvrzení asynchronního uložení

změn. V případě kvalitní implementace nedochází k výraznému zatížení zdrojového sytému a k větším odezvám aplikace. Také tím, že se o tuto komunikaci stará speciální vrstva aplikace, tak ve většině případů nedochází k větší pracnosti případného dalšího vývoje nebo úprav aplikace, protože vývojáři jsou od tohoto řešení odstíněni.

Tento přístup má také své nevýhody. Již při nákupu nebo vývoji nové aplikace se musí počítat s požadavkem na tuto funkcionalitu. V případě selhání asynchronního zaslání a uložení změn musí existovat způsob odchycení těchto chyb a také postup, jak danou situaci řešit.

Asi největší nevýhodou tohoto řešení je, že nezachytí změny provedené přímo v databázi. Může se jednat například o zásah databázového administrátora v případě nestandardních situací. Další výraznou nevýhodou je, že v případě použití tohoto způsobu získávání změn do datového skladu je typicky potřeba řešit získání dat z více zdrojových systémů. To vede k tomu, že tento způsob řešení buď musí umožňovat všechny zdrojové aplikace, což je velmi nepravděpodobné vzhledem k tomu, že zdrojové aplikace jsou často starší než datový sklad, nebo to vede k použití různých způsobů získání dat u různých zdrojových systémů, což značně zesložiťuje architekturu řešení a tím i jeho správu a provoz.

6.2.2 Využití funkcionality zdrojových databází

Další možností získání dat ze zdrojových systémů je využití funkcionalit zdrojových databází. Princip tohoto řešení je velmi podobný předchozímu, toto řešení se ale nenachází na úrovni aplikace, ale je vytvořeno přímo v databázi zdrojového systému.

Často se jedná o různé Triggery, tedy zdrojové kódy spustitelné v databázi, které se aktivují při zvolených databázových operacích. V případě zachytávání změn v datech se jedná především o operace typu Insert, Update a Delete.

Tento zdrojový kód se poté postará o zaznamenání změny a provede export této změny do jiné tabulky nebo souboru. Tato tabulka nebo soubor se v praxi často nazývá jako rozdílová tabulka/soubor nebo delta tabulka/soubor. Důležité je mít na paměti, že jako v předchozím případě tyto tabulky nebo soubory obsahují pouze změny provedené ve zdrojových systémech a nikoli celou aktuální množinu dat.

Výhodou tohoto řešení oproti předchozímu je především to, že je schopné zachytit i změny provedené přímo v databázi (například i změnu provedenou databázovým administrátorem přímo nad databází při nestandardních situacích).

Nevýhodou může být potencionální větší zátěž zdrojové databáze, zvláště v případě použití Triggeru, který zapisuje do delta tabulky. V tomto případě může být zátěž této databáze až dvojnásobná, protože každá změna musí být promítnuta dvakrát, jednou do původní tabulky a jednou do delta tabulky.

Při tomto řešení také zůstává ta nevýhoda, že u každého zdrojového sytému je potřeba zajistit získání dat jednotlivě. V případě, kdy zdrojové systémy používají různé databáze, znamená to vývoj, otestování a nasazení kódů a celého řešení na každou takovou databázi zvlášť. To může výrazně prodražit celé řešení a značně zkomplikovat jeho udržitelnost v dlouhodobém horizontu.

6.2.3 Export dat

Na první pohled nejjednodušší možností jak získat data ze zdrojových systémů je jejich export z databáze. Tento export může být proveden pomocí databázového nástroje, který je součástí dané databáze nebo je dodáván spolu s databází. Může být také proveden ETL nástrojem.

V obou případech je velmi komplikované exportovat pouze data, která se změnila od posledního exportu. Důvodem je, že rozpoznání těchto dat je problémové. Z tohoto důvodu jsou někdy v zdrojové databázi použity sloupce, které identifikují záznamy, které se od posledního exportu změnily. Často tyto sloupce nesou názvy jako „datum_modifikace“, „audit_sloupec“, „timestamp“ a podobně.

Problém tohoto řešení je v jeho složitosti. Při každé změně v datech se musí hodnota tohoto sloupce také vhodně změnit. Každá zdrojová aplikace, která s těmito daty pracuje, musí s touto povinností počítat. To samé platí i pro ruční zásahy do dat v případě nestandardních situací.

Vzhledem k tomu, že tento princip je potřeba zajistit ve všech zdrojových databázích a ve všech jejich strukturách, stává se celé řešení velice náročné na provoz a případné rozšiřování. Otázkou je také důvěra v takováto řešení, zejména to, že všechny operace se

zdrojovými daty správně modifikují určený sloupec. Například malá chyba v drobném rozšíření zdrojového systému může způsobit, že změny z určité části zdrojového systému nebudou zachyceny. Cílový systém, jakým je například právě datový sklad, ale obvykle nemá prostředky jak tuto chybu rychle odhalit. S odstupem času se také odhalení takovýchto chyb stává velmi složitým.

Výše uvedené důvody většinou vyústí ke zvolení daleko jednoduššího řešení, jakým je celkový export dat. Při celkovém exportu dat dochází k získání všech relevantních dat ze zdrojových systémů a jejich nahrání do cílového systému. Je pak na cílovém systému aby rozlišil nová data, změněná data a smazaná data a vhodně je zpracoval.

Výhodou tohoto řešení je poměrně jednoduché provedení, které lze uskutečnit bez změn v logice zdrojových systémů. Toto řešení je také poměrně robustní ve smyslu odolnosti k chybám. Tím, že dochází k celkovému exportu dat, je velmi nízké riziko nepodchycení veškerých změn vzniklých ve zdrojových systémech.

Nevýhodou tohoto řešení je značné vytížení zdrojových systémů v době exportu dat. Proto se v případě použití tohoto řešení často provádí tento export v době, kdy jsou zdrojové systémy méně vytěžované, obvykle během nočních hodin. To má za důsledek, že aktuálnost dat v cílovém systému je dána tím, kdy jsou zdrojové systémy k dispozici pro tento celkový export.

V případě exportu během nočních hodin jsou data cílovému systému dodána se zpožděním jednoho dne. U nadnárodních společností, které operují přes více časových pásem, může vzniknout stav, kdy nelze nalézt žádné časové okno vhodné pro export dat.

Pokud se zvolí cesta celkového exportu dat a najde se pro něj vhodné časové okno, je stále toto řešení značně neefektivní. Je totiž velice nepravděpodobné, že by zdrojové systémy měnily mezi jednotlivými exporty veškerá svá data. Je možné, že s veškerými daty pracují, například je používají pro čtení a vhodné zobrazení uživateli, obvykle ale mění jen malou část svých dat.

V reálném prostředí lze poměrně dobře odhadnout (například podle dvou po sobě jdoucích exportech) procentuální změnu ve zdrojových datech. V případě, kdy by se zjistilo, že se ve zdrojovém systému mění pouze 5 % všech dat mezi jednotlivými celkovými exporty a

datová základna tohoto zdrojového systému by byla 200 GB, dochází ke zbytečnému přenosu 190 GB dat.

Nejedná se tu jen o samotnou náročnost přenosu dat, ale o celý proces s tím související. Dochází totiž ke zbytečnému vytížení zdrojového systému, který musí těchto 190 GB poskytnout, poté následuje přenos těchto dat po infrastruktuře a následně k samotnému nahrání dat do cílového systému. Cílový systém je poté ještě vytížen tím, že musí data vhodně zpracovat, tedy ve většině případů, je na něj přesunuta poměrně výpočetně náročná úloha, a to zjistit, která data z oněch celkových 200 GB byla změněna a jsou nezbytná k dalšímu zpracování.

Čím je potřeba mít data aktuálnější v cílovém systému, tím častěji při použití tohoto řešení dochází ke zbytečnému vytěžování všech tří prvků, tedy zdrojového systému, infrastruktury a cílového systému. Z tohoto důvodu se nejeví toto řešení jako vhodné k běžnému použití, zvláště pak v prostředích, kde zdrojové systémy mají velké datové základny.

6.2.4 Specializovaný CDC nástroj

Principem fungování specializovaného CDC nástroje je schopnost vyčítat provedené změny v databázích z jejich databázových logů. Ve většině případů tedy není potřeba nahrávat data z těchto databází a z jejich diskových úložišť, ale pomocí znalosti databázových logů lze odvodit, jaká data byla jak změněna a na základě toho vyvodit potřebný stav v cílovém uložišti.

Tímto způsobem se lze vyhnout výraznému zatížení zdrojových systémů a zároveň je možné potřebná data zpřístupnit pro další zpracování. Za další výhodu může být považováno relativně rychlé zachycení změn ve zdrojových systémech, a tedy tyto změny mohou být velmi rychle zapisovány do cílového systému a zpřístupněny tak k dalšímu zpracování.

Tento způsob získání datových změn ze zdrojových systémů na první pohled nepřináší výrazné nevýhody, tak jako předchozí uvedené možnosti. I z tohoto důvodu byl tento způsob získávání dat zvolen k vyzkoušení v testovacím prostředí, kdy bude kladen důraz

na odhalení možných nevýhod, které by mohly vzniknout v rámci jeho nasazení v reálném prostředí datového skladu.

V případě nasazení specializovaného CDC nástroje v konkrétním prostředí lze spatřit velkou výhodu zejména v případě, kdy zvolený nástroj umí změny získávat ze všech zdrojových systémů, které jsou provozované v daném prostředí. V tomto případě by nasazení tohoto specializovaného nástroje mohlo vést ke zjednodušení používaného řešení, kdy místo různých způsobů získávání dat ze zdrojových systémů by existoval jednotný způsob získávání zdrojových dat. Správa takového řešení a jeho provoz by tak mohl být efektivnější.

6.3 IBM InfoSphere Change Data Capture

Pro vyzkoušení tohoto konceptu v testovacím prostředí byl zvolen produkt IBM InfoSphere Change Data Capture, který je od společnosti IBM. Tento produkt je součástí řešení IBM InfoSphere Data Replication a nabízí již zmíněné výhody specializovaného CDC nástroje. Zejména se jedná o získání dat s nízkou zátěží zdrojového systému, rychlé získání změn ze zdrojového systému a relativní jednoduchost nasazení v heterogenním prostředí s různými podporovanými systémy (IBM, 2016b).

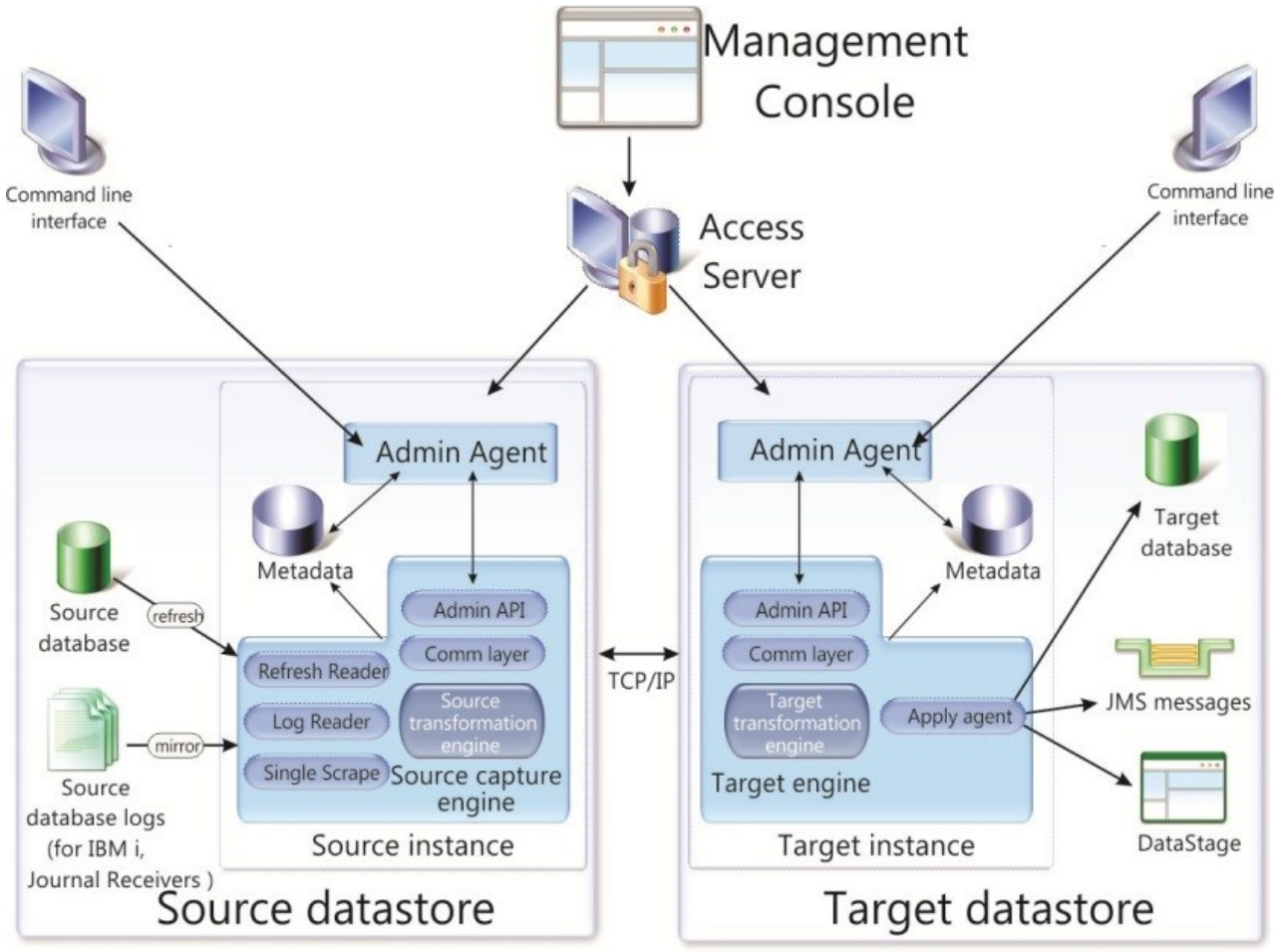
Tento nástroj podporuje různé zdrojové a cílové systémy. Nejedná se přitom pouze o klasické relační databáze. V seznamu podporovaných systémů lze nalézt i platformu Hadoop pro Big Data, ETL nástroj DataStage a middleware Event Server pro správu událostí. Jejich přehled je uveden v následující tabulce (Tabulka 1), která přehledně znázorňuje podporované zdrojové a cílové systémy.

Tabulka 1: Podporované zdrojové a cílové systémy produktem IBM InfoSphere Change Data Capture (IBM, 2016c)

Podporované zdrojové systémy	Podporované cílové systémy
IBM DB2 for Linux, UNIX and Windows (LUW)	IBM DB2 for Linux, UNIX and Windows (LUW)
IBM DB2 for i	IBM DB2 for i
IBM DB2 for z/OS	IBM DB2 for z/OS
IBM Informix Dynamic Server	IBM Informix Dynamic Server
IMS	CDC Replication Engine for Event Server
Microsoft SQL Server	IBM InfoSphere DataStage
Oracle	Microsoft SQL Server
Sybase	Netezza
	Oracle
	Sybase
	Teradata
	CDC Replication Engine for FlexRep
	IBM Cloudant
	Apache Hadoop

Hlavní komponenty produktu IBM InfoSphere Change Data Capture jsou zobrazeny na následujícím obrázku (Obr. 4). Zejména klíčové komponenty tohoto produktu jsou:

- Management Console,
- Access Server,
- Source datastore
- Target datastore.



Obr. 4 – Architektura nástroje IBM InfoSphere Change Data Capture (IBM, 2016d)

Management Console je tlustý klient, který uživateli umožňuje konfigurovat, monitorovat a řídit získávání dat z různých systémů a jejich přesun do cílových systémů. Management Console umožňuje sledovat různé statistiky o získávání změn ze zdrojových systémů, včetně doby prodlení mezi získáním dat ze zdrojového systému a jejich zápisem do cílového systému. Jak je z obrázku patrné, Management Console komunikuje s Access Serverem.

Access Server je primární řídící bod, který přijímá příkazy z Management Console. Access Server vykonává příkazy z Management Console a obsahuje patřičná nastavení, která byla zvolena pro získání dat ze zdrojových systémů a jejich uložení do cílových systémů. Uživatel, který pracuje s CDC přes Management Console, a provede patřičná nastavení, může poté Management Console zavřít bez dopadů na přenosy dat mezi systémy.

Source datastore a Target datastore jsou instance CDC, které obsahují datové soubory a samotný replikační engine. Z pohledu uživatele každý datastore reprezentuje databázi, ke které se CDC připojuje. Datastores jsou tedy určitými kontejnery, které umožňují pracovat se zdrojovými a cílovými systémy (IBM, 2016d).

Obvyklý průběh nastavení a použití CDC je následující. Nejdříve je potřeba nainstalovat Access Server a Management Console. Poté je potřeba nainstalovat a správně nakonfigurovat jednotlivé datastore na každém zdrojovém nebo cílovém systému, který bude použit v rámci CDC. Následně je potřeba vytvořit tzv. Subscriptions, která obsahují spojení nezbytné k přenosu dat mezi zdrojovým a cílovým datastore. Tyto Subscriptions obsahují detaily o datech, která jsou přenášena a také nastavení, jak mají být zdrojová data uložena do cílového systému.

V momentě, kdy jsou Subscriptions vytvořena je nutné nastavit mapování jednotlivých tabulek. Mapování se provádí obdobně jako při použití klasických ETL nástrojů, obvykle tedy, který sloupec zdrojové tabulky se propisuje do kterého sloupce cílové tabulky. V případě použíti CDC pro Event Server se vytváří a modifikují XML zprávy, které obsahují mapování ze zdrojových systémů. V případě použití ETL nástroje Data Stage, jako cílového systému (často pro potřeby dalšího rychlého zpracování dat) se generují definiční soubory typu „.dsx“, případně další Java třídy pro Data Stage jobs.

Posledním krokem při používání CDC je vlastní spouštění a zastavování datových replikací, tedy spouštění a zastavování vlastního přenosu dat ze zdrojových systémů do cílových systémů. Toto řízení replikací může být prováděno ručně nebo i pomocí automatizačních skriptů, které využívají připravené aplikační rozhraní (API).

6.4 Nasazení v testovacím prostředí

Tato subkapitola popisuje, jak koncept CDC byl začleněn do testovacího prostředí, tak aby bylo umožněno jeho otestování. Jako zdrojová databáze byla použita databáze DB2 for LUW, která představovala zdrojový systém. DB2 byla vybrána, protože je vhodná pro transakční systémy, které jsou vytížené mnoha kratšími (často souběžnými) dotazy, které je potřeba vykonávat s co nejmenší časovou odezvou. Pro testování byla vybrána poslední

vydaná verze této databáze dostupná v době psaní této diplomové práce, tedy DB2 verze 11.1 pro Windows.

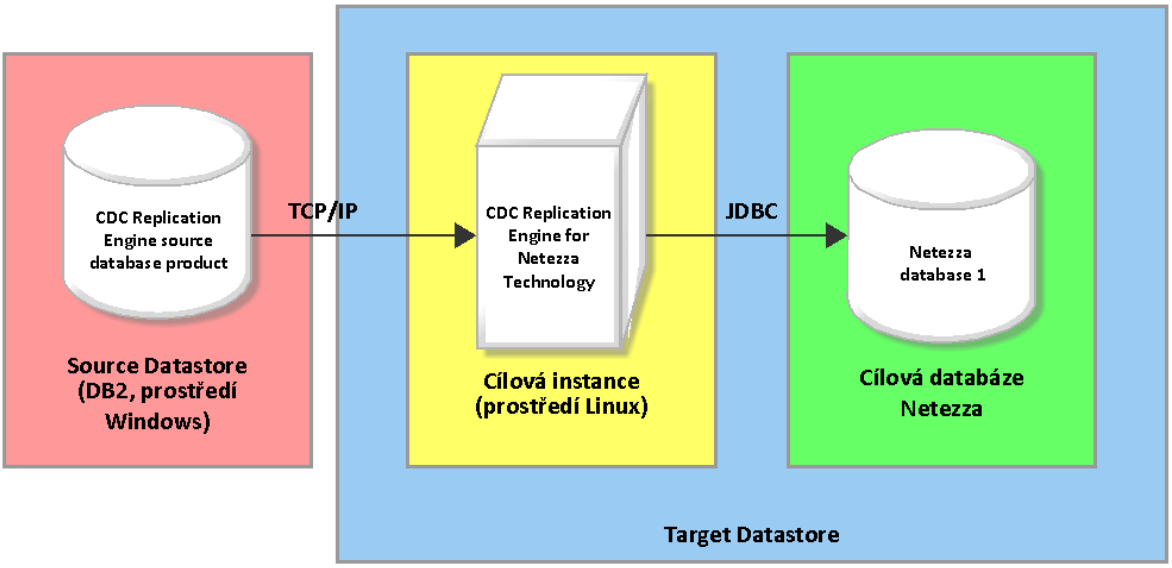
V prostředí Windows byl dále naistalován Access Server a Management Console. V prostředí Windows na straně zdrojového systému DB2 byl dále nainstalován CDC Replication Engine for DB2, který byl následně upgradován na poslední dostupnou verzi 11.3.3.3. Spolu s DB2 tento Engine tvoří Source datastore, tak je jak zobrazeno na obrázku č. 4.

Jako cílový systém byla zvolena databáze Netezza. Tato databáze je vhodná pro náročné analytické dotazy a je již detailněji popsaná v kapitole 5. Testovací prostředí a technologie. V prostředí Linux byl následně instalován Replication Engine for IBM Netezza, který tvoří cílovou instanci a spolu s databází tvoří Target datastore.

Tato cílová instance také nabízí funkcionalitu, která umožňuje překlenout technologické rozdíly, například obsahuje dočasné uložiště, které odstíní Netezzu od velmi rychlých a často souběžných transakcí, které by jí zbytečně vytěžovaly. Cílová instance poté změny do Netezzy promítne v dávce, tedy ve formě, ke které je Netezza uzpůsobená.

Pomocí nastavení lze tedy optimalizovat zatížení cílové databáze a určitého zpoždění, s kterým budou změny ze zdrojového systému promítnuty do cílového systému. Toto nastavení je realizované pomocí parametru cílové instance a v rámci testovacího prostředí byl tento parametr nastaven na jednu minutu.

Následující obrázek č. 5 přehledně ilustruje toto propojení. Základní tři komponenty jsou na obrázku znázorněny podle dokumentace (IBM, 2016e), kdy Source Datastore komunikuje s cílovou instancí přes TCP/IP, tato instance poté promítá změny do cílové databáze pomocí JDBC. Podbarvení obrázku s vysvětlujícími poznámkami je vytvořeno na základě nově vytvořeného testovacího prostředí, tedy se jedná o konkrétní provedení v testovacím prostředí.



Obr. 5 – Propojení Source Datastore a Target Datastore v testovacím prostředí zpracováno dle dokumentace (IBM, 2016) a dle testovacího prostředí (zdroj: autor)

6.5 Cíle a zvolený postup vyzkoušení konceptu v testovacím prostředí

Tato subkapitola popisuje cíle a výstupy, kterých by mělo být dosaženo při vyzkoušení tohoto konceptu v rámci testovacího prostředí. Tyto cíle a výstupy již byly krátce zmíněny v kapitole 4.1 Zachycení změn ve zdrojových systémech, tedy v rámci zdůvodnění volby tohoto rozšiřujícího konceptu.

Cílem a výstupem vyzkoušení tohoto rozšiřujícího konceptu v rámci testovacího prostředí by mělo být zjištění, jak a v jaké formě mohou být změněná data dostupná v cílovém systému a tedy jak s nimi bude možné dále pracovat pro potřeby datového skladu. Jedná se zejména o možnosti dalšího zpracování těchto dat tak, aby bylo možné efektivně udržovat historii dat v datovém skladu v momentě, kdy datový sklad dostává průběžně pouze změněná data.

V rámci vyzkoušení tedy bude vytvořena sada testovacích dat, pomocí které bude simulována práce na zdrojovém systému. Na cílovém systému poté bude sledováno, jak jsou změny zachyceny, případně pomocí vhodné konfigurace CDC bude měněn způsob

ukládání změn v cílovém systému. Na základě tohoto testu poté bude vyvozen možný dopad do datového skladu a také to, jaké se nabízí další použití či zpracování takto zaznamenaných dat.

Během nasazení tohoto rozšiřujícího konceptu a během jeho celého testování bude také kladen důraz na odhalení případných rizik, které by mohly mít dopady při použití v produkčních prostředích.

6.6 Vyzkoušení konceptu v testovacím prostředí

Tato kapitola obsahuje postup a popis různého nastavení, použitého při vyzkoušení konceptu v testovacím prostředí. V následujících odstavcích je krátce popsán společný základ použitý v testovacím prostředí. Následují dvě subkapitoly, které obsahují klíčové možnosti nastavení přenosu změn mezi zdrojovým a cílovým systémem. Poté je uvedena kapitola se samotným postupem v testovacím prostředí, která obsahuje výstupy z tohoto testovacího prostředí.

Pro vyzkoušení konceptu v testovacím prostředí bylo vytvořeno několik tabulek ve zdrojové databázi DB2. Tyto tabulky byly naplněny daty tak, aby obsahovaly určitý stav ve zdrojovém systému před připojením na CDC. Následně byly v Management konzoli připojeny nakonfigurované Source Datastore (DB2) a Target Datastore (Netezza).

Po připojení obou datastore byla vytvořena subscriptions, která obsahovala tyto datastore. Do této subscription bylo dle potřeb různých testů přidáváno mapování tabulek. Při definici mapování tabulky se volí dvě nastavení přenosu dat, která jsou z pohledu datového skladu pro další zpracování velmi důležitá. První z nich je metoda přenosu dat a druhou z nich je typ mapování. Jelikož obě nastavení jsou zcela klíčová pro návazné zpracování dat, jsou jim věnovány další dvě podkapitoly.

6.6.1 Metody přenosu dat

Při vytváření mapování tabulky se volí ze dvou možností Mirror (v češtině zrcadlení) nebo Refresh (také pod názvem Snapshot, v češtině se lze setkat také s pojmenováním snímek).

Metoda zrcadlení používá klasický způsob CDC pro získávání změn a to pomocí databázových logů. V momentě, kdy je tato metoda použita, CDC agent čte databázové logy a nalezne-li změnu v daných tabulkách, zasílá tuto informaci do cílového systému. Nevýhodou této metody je, že nedokáže detekovat data, která již v databázi jsou delší dobu uložená. Přesněji řečeno se jedná o taková data, která již nejsou k dispozici v aktuálně dostupných databázových lozích.

Naproti tomu metoda snímku se nespoléhá na databázové logy, ale používá přístup přímo do databáze k načtení všech dat. Tím je garantováno získání všech dat ze zdroje k danému momentu načtení dat. S touto metodou jsou ale spojené klasické nevýhody tohoto způsobu získávání dat, zejména se jedná o vytížení zdrojové databáze.

Ideální volbou pro praktické použití je potom kombinace obou metod. Jedná se o nastavení zrcadlení (Mirror), kdy je požadován snímek (Refresh) před spuštěním zrcadlení. Tato volba je také známá pod názvem Refresh before Mirror. Touto metodou lze tedy docílit získání všech dat ze zdrojového systému a poté zaznamenávat pouze změny ve zdrojovém systému s minimálním výkonovým dopadem.

6.6.2 Typy mapování

Typ mapování určuje, jak budou data propisována do cílového systému. Je na výběr ze tří hlavních možností Standard, LiveAudit a Adaptive Apply. Tyto tři možnosti jsou zde krátce popsány a poté i vyzkoušeny na testovací tabulce v následující podkapitole.

Typ mapování Standard propisuje změny do cílové tabulky tak, jak byly prováděny ve zdrojovém systému. Pokud do zdrojového systému byla data vložena, jsou také vložena do cílového systému. Pokud byla data smazána, jsou data smazána i v cílovém systému.

Typ mapování LiveAudit umožňuje sledovat všechny změny, které byly provedeny ve zdrojovém systému. Každá operace, která byla provedena ve zdrojovém systému, způsobí přidání záznamu do cílové tabulky včetně značky, která určuje provedenou změnu.

Typ mapování Adaptive Apply lze použít v případech, kdy zdrojová a cílová tabulka není synchronizována a přesto je potřeba přenášet data ze zdrojového systému do cílového. Například pokud záznam v cílovém systému neexistuje ale ve zdrojovém ano a byl na něm

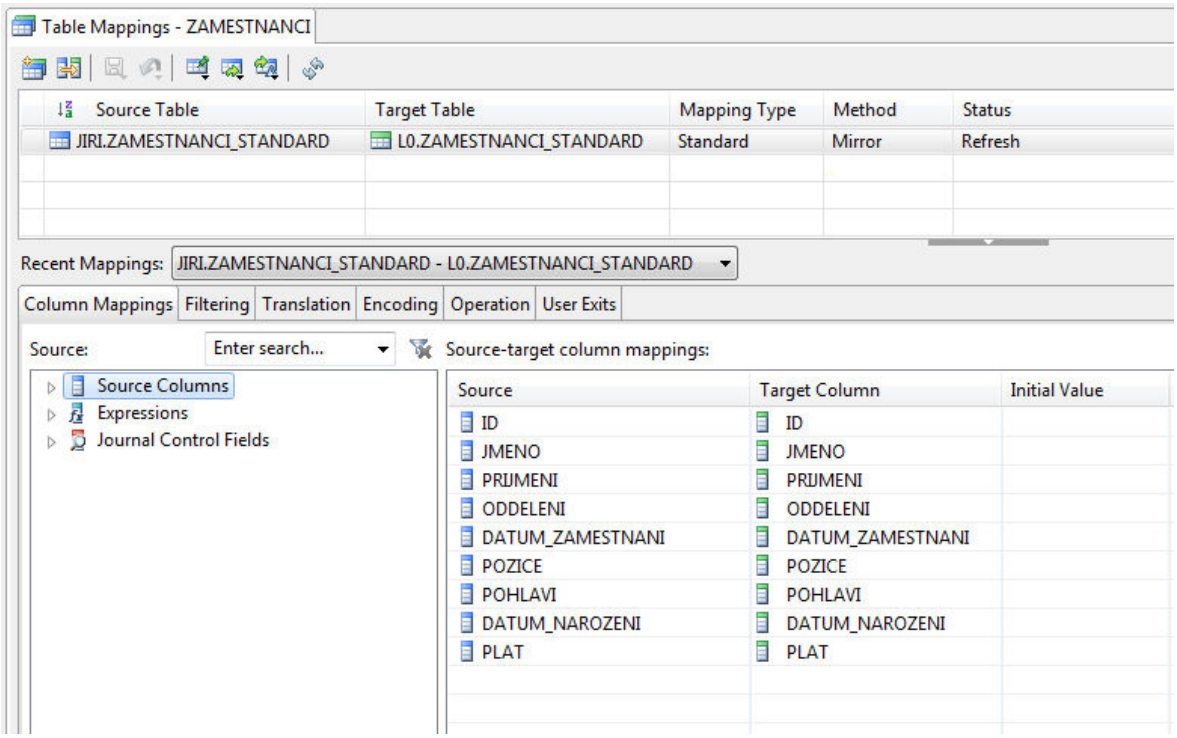
provedena změna (update), potom tento typ mapování v cílovém systému provede vložení nového záznamu.

6.6.3 Postup v testovacím prostředí a výstupy z testovacího prostředí

Všechny tři uvedené typy mapování a obě dvě metody přenosu dat byly postupně vyzkoušené v testovacím prostředí. Vzhledem k rozdílným výstupům na straně cílového systému byl kladen hlavní důraz na možnosti dalšího zpracování dat v cílovém systému při použití různých nastavení CDC.

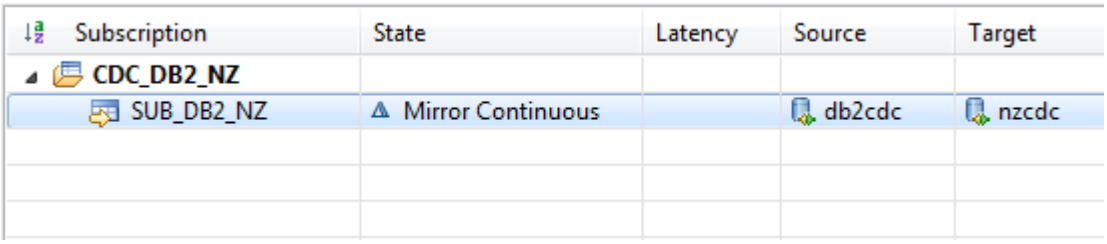
6.6.3.1 Mapování Standard

Do předpřipravené subscription, která je popsána v kapitole 6.6 Vyzkoušení konceptu v testovacím prostředí, bylo přidáno mapování tabulky „zamestnanci_standard“. Tato tabulka existuje v DB2 ve schématu JIRI a obsahuje 42 záznamů. V Netezze byla vytvořena tato tabulka ve schématu L0 a obsahuje 0 záznamů. Typ mapování byl nastaven na standard a zvolená metoda přenosu dat byla zvolena na mirroring. Jelikož se jedná o nově přidanou tabulku k přenášení, produkt IBM InfoSphere Change Data Capture automaticky změnil stav mappingu na refresh, tedy vynucení pořízení snímku před samotným mirroringem (tzv. Refresh before Mirror). Tím byla daná tabulka připravena k použití v rámci CDC. Tento stav je zobrazen na obrázku číslo 6.



Obr. 6 – Připravená tabulka zamestnanci_standard k přenášení dat v rámci CDC Mapování Standard (zdroj: autor)

Subscription s daným mapováním je po tomto kroku možné spustit. Objeví se upozornění, že i přesto, že je nastaven mirroring je potřeba provést prvotní refresh celé tabulky. Po potvrzení volby, se provede refresh, který je bezprostředně následován mirroringem. Tato situace je zobrazena na obrázku číslo 7. Se zapnutým mirroringem je poté na zdrojovém systému provedena skupina příkazů, které manipulují s daty. Jednalo se o operace typu INSERT, UPDATE, DELETE a TRUNCATE. Obrázek číslo 8 zobrazuje stav zdrojové tabulky po provedení těchto příkazů ve zdrojovém systému DB2. Obrázek číslo 9 zobrazuje stav cílové tabulky po provedení příkazů v cílovém systému Netezza.



Obr. 7 – Zobrazení aktivní subscription v Management Console (zdroj: autor)

Properties | SQL Results

	ID	JMENO	PRIJMENI	ODDELENI	DATUM_ZAMESTNANI	POZICE
1	1	Jan	Novák	A00	2013-04-26	Analytik
2	2	Jiří	Cipísek	A10	1991-01-01	MANAGER
3	3	Adam	Klaun	B00	2014-09-18	Analytik
4	4	Karel	Žlutý	A10	2015-07-17	Analytik
5	5	Tereza	Pokorná	A10	2013-08-25	MANAGER
6	6	Lukáš	Dlouhý	B00	2015-12-04	Intern
7	7	Michal	Hula	B00	2014-10-19	Tester
8	8	Milan	Konvalinka	A00	2015-01-20	Tester
9	9	Martin	Modrý	B00	1991-01-01	Tester

Fetched 51 records, 51 records shown

Obr. 8 – Stav tabulky ve zdrojovém systému DB2 po provedení příkazů při Mapování Standard (počet řádků i sloupců je zkrácen) (zdroj: autor)

Output | Result 1

Drag a column header here to group by that column.

	ID	JMEN	PRIJMENI	ODDELENI	DATUM_ZAMESTNANI	POZICE
1	1	Jan	Novák	A00	2013-04-26 00:00:00	Analytik
2	2	Jiří	Cipísek	A10	1991-01-01 00:00:00	MANAGER
3	3	Adam	Klaun	B00	2014-09-18 00:00:00	Analytik
4	4	Karel	Žlutý	A10	2015-07-17 00:00:00	Analytik
5	5	Tereza	Pokorná	A10	2013-08-25 00:00:00	MANAGER
6	6	Lukáš	Dlouhý	B00	2015-12-04 00:00:00	Intern
7	7	Michal	Hula	B00	2014-10-19 00:00:00	Tester
8	8	Milan	Konvalinka	A00	2015-01-20 00:00:00	Tester
9	9	Martin	Modrý	B00	1991-01-01 00:00:00	Tester

1 query, 0,070 sec., 51 rows * sql_for_cdc_new_db_and_tables.sql

Obr. 9 – Stav tabulky v cílovém systému Netezza po provedení příkazů při Mapování Standard (počet řádků i sloupců je zkrácen) (zdroj: autor)

Jak je z obrázků patrné, změny ve zdrojovém systému se v cílovém systému propisují zcela identicky. Uživatel, který přistupuje do cílového systému, tedy může s daty pracovat stejně, jako kdyby přistupoval do zdrojového systému.

6.6.3.2 Mapování LiveAudit

Obdobně jako v předchozí kapitole bylo přidáno nové mapování tabulky, pro vyzkoušení typu mapování LiveAudit. Na straně zdrojového systému byla použita nová tabulka „zamestnanci_LA“ o stejné struktuře a s podobnými daty jako tabulka použitá v předchozí kapitole. Nová tabulka byla také vytvořena v cílovém systému Netezza. Základní struktura tabulky byla stejná jako tabulka zdrojového systému. Pro potřeby mapování LiveAudit byla cílová tabulka ještě rozšířena o technické sloupce, které obsahují další popisné informace o významu daného řádku.

Tato technická pole mohou plnit různé úlohy. Velké množství těchto polí je již v samotném produktu předdefinováno a lze je nalézt v dokumentaci (IBM, 2016f). Pro potřeby testu a následného použití dat pro další možnosti zpracování byla vybrána čtyři technická pole:

- Entry Type (&ENTTYP)
- Record Modification Time (&Timestamp)
- Record Modification User (&USER)
- Commit Cycle ID (&CCID)

Entry Type indikuje význam daného řádku, nese tedy informaci o typu změny na zdrojovém systému. Record Modification Time je časová známka změny ve zdrojovém systému. Record Modification User je jméno uživatele, který provedl změnu ve zdrojovém systému. Commit Cycle ID je číslo transakce, která provedla změnu ve zdrojovém systému.

Z pohledu cílového systému a zejména z pohledu datového skladu je velmi důležité vědět, v jakém pořadí byly změny na zdrojovém systému provedeny. Jedná se například o situaci, kdy jeden záznam byl několikrát za sebou změněn. Pro cílový systém je velmi důležitá informace, která změna proběhla jako poslední a tedy jaký je konečný stav daného záznamu.

Z technických polí se k tomuto použití nabízí pole Record Modification Time (časová známka) a Commit Cycle ID (číslo transakce). Časová známka v tomto případě nemusí být dostatečná. Zvláště na vytížených produkčních systémech nemusí podporovaná nejmenší jednotka času mít dostatečnou granularitu a může tak proběhnout několik změn na zdrojovém systému v rámci této jednotky.

Z testů také vyplynulo, že ani číslo transakce není v tomto případě dostatečné. V rámci jedné transakce může totiž proběhnout změna stejného řádku několikrát. V tento moment by opět nebylo možné na straně cílového systému určit výsledný stav daného řádku.

K řešení této situace lze využít vlastnost CDC, která zajišťuje, že změny jsou zasílány do cílového systému v takovém pořadí, v jakém proběhly na zdrojovém systému. V případě použití CDC ve spojení s ETL nástrojem DataStage je možné například tyto změny sekvenčně číslovat v rámci jednotlivých DataStage jobů. V případě testovacího prostředí byla použita Java třída, která příchozí řádky sekvenčně vzestupně čísluje.

Toto pole bylo také přidáno do cílové tabulky a do mapování definující převod mezi zdrojovou a cílovou tabulkou. Zvolená metoda přenosu dat byla nastavena na mirroring s prvotním refreshem (tzv. Refresh before Mirror).

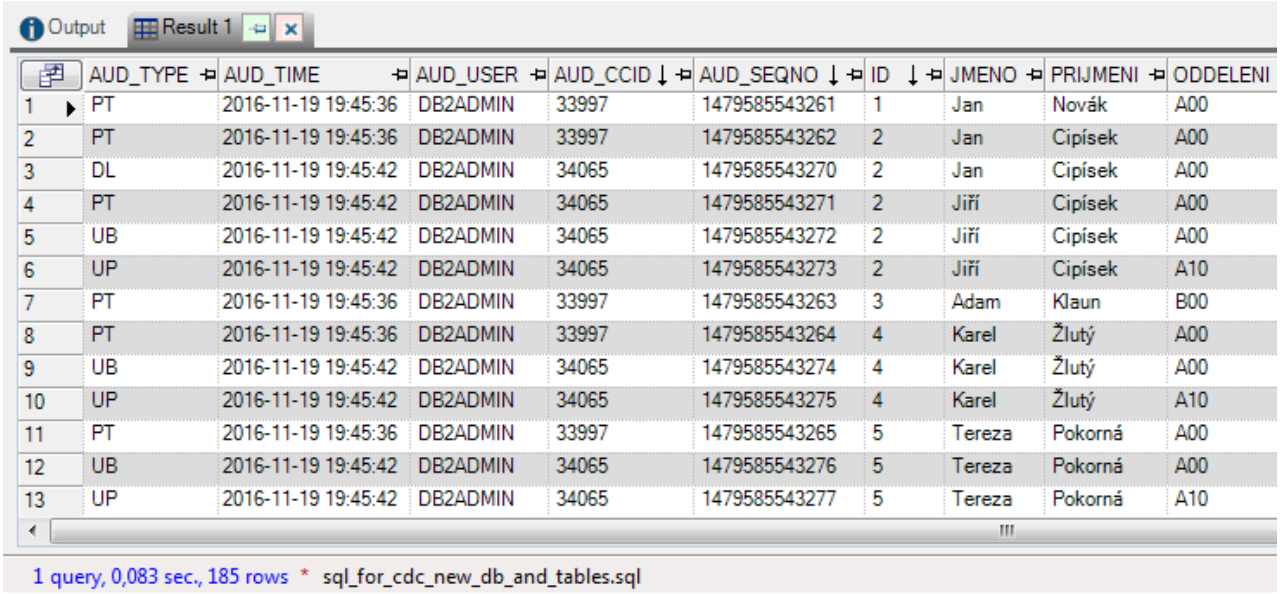
Po tomto kroku je možné subscription s tímto mapováním spustit. Jako v předchozí subkapitole je následně provedena na zdrojovém systému skupina příkazů, které manipulují s daty. Opět byly použity operace typu INSERT, UPDATE, DELETE a TRUNCATE. Obrázek číslo 10 zobrazuje stav tabulky ve zdrojovém systému DB2 po provedení těchto příkazů. Obrázek číslo 11 zobrazuje stav tabulky po provedení příkazů v cílovém systému Netezza.

Properties | SQL Results

	ID	JMENO	PRIJMENI	ODDELENI	DATUM_ZAMESTNANI	POZICE
1	1	Jan	Novák	A00	2013-04-26	Analytik
2	2	Jiří	Cipísek	A10	1991-01-01	MANAGER
3	3	Adam	Klaun	B00	2014-09-18	Analytik
4	4	Karel	Žlutý	A10	2015-07-17	Analytik
5	5	Tereza	Pokorná	A10	2013-08-25	MANAGER
6	6	Lukáš	Dlouhý	B00	2015-12-04	Intern
7	7	Michal	Hula	B00	2014-10-19	Tester
8	8	Milan	Konvalinka	A00	2015-01-20	Tester
9	9	Martin	Modrý	B00	1991-01-01	Tester

Fetched 51 records, 51 records shown

Obr. 10 – Stav tabulky ve zdrojovém systému DB2 po provedení příkazů při Mapování LiveAudit (počet řádků i sloupců je zkrácen) (zdroj: autor)



Output | Result 1

	AUD_TYPE	AUD_TIME	AUD_USER	AUD_CCID	AUD_SEQNO	ID	JMENO	PRIJMENI	ODDELENI
1	PT	2016-11-19 19:45:36	DB2ADMIN	33997	1479585543261	1	Jan	Novák	A00
2	PT	2016-11-19 19:45:36	DB2ADMIN	33997	1479585543262	2	Jan	Cipísek	A00
3	DL	2016-11-19 19:45:42	DB2ADMIN	34065	1479585543270	2	Jan	Cipísek	A00
4	PT	2016-11-19 19:45:42	DB2ADMIN	34065	1479585543271	2	Jiří	Cipísek	A00
5	UB	2016-11-19 19:45:42	DB2ADMIN	34065	1479585543272	2	Jiří	Cipísek	A00
6	UP	2016-11-19 19:45:42	DB2ADMIN	34065	1479585543273	2	Jiří	Cipísek	A10
7	PT	2016-11-19 19:45:36	DB2ADMIN	33997	1479585543263	3	Adam	Klaun	B00
8	PT	2016-11-19 19:45:36	DB2ADMIN	33997	1479585543264	4	Karel	Žlutý	A00
9	UB	2016-11-19 19:45:42	DB2ADMIN	34065	1479585543274	4	Karel	Žlutý	A00
10	UP	2016-11-19 19:45:42	DB2ADMIN	34065	1479585543275	4	Karel	Žlutý	A10
11	PT	2016-11-19 19:45:36	DB2ADMIN	33997	1479585543265	5	Tereza	Pokorná	A00
12	UB	2016-11-19 19:45:42	DB2ADMIN	34065	1479585543276	5	Tereza	Pokorná	A00
13	UP	2016-11-19 19:45:42	DB2ADMIN	34065	1479585543277	5	Tereza	Pokorná	A10

1 query, 0,083 sec., 185 rows * sql_for_cdc_new_db_and_tables.sql

Obr. 11 – Stav tabulky v cílovém systému Netezza po provedení příkazů při Mapování LiveAudit (počet řádků i sloupců je zkrácen) (zdroj: autor)

Na obrázku číslo 11 je vidět všech pět technických sloupců. Zejména důležitý je první sloupec, který, jak již bylo uvedeno, obsahuje typ změny na zdrojovém systému. Význam jeho jednotlivých zkratek lze nalézt v dokumentaci (IBM, 2016f). Například na obrázku zobrazené PT je zkratka, která indikuje INSERT, tedy že daný řádek s těmito hodnotami byl vložen do zdrojového systému. Zkratka DL indikuje DELETE daného řádku. Zkratka UB indikuje stav řádku před UPDATE a zkratka UP stav řádku po UPDATE. Pomocí těchto zkratek lze tedy přesně sledovat, jaké operace byly na zdrojovém systému provedeny.

V druhém sloupci je časová známka změny na zdrojovém systému. V třetím sloupci je uživatelský účet zdrojového systému, který provedl změnu. Čtvrtý je číslo transakce zdrojového systému, která provedla danou změnu.

Například na řádku číslo 3 až 6 je uvedeno stejné číslo transakce 34065 pro změny záznamu ve zdrojovém systému s ID = 2. Z toho vyplývá, že všechny změny tohoto záznamu na zdrojovém systému byly provedeny v rámci jedné transakce, která byla ukončena příkazem COMMIT. V rámci této transakce proběhlo smazání daného řádku (DL), vložení nového řádku (PT), a změna daného řádku (UB a UP). Jak je vidět z tohoto příkladu, čistě na základě čísla transakce nelze vyvodit výsledný stav zdrojové tabulky.

Pomocí pátého sloupce, který obsahuje sekvenční číslo tak, jak byly změny prováděny na zdrojovém systému lze i v rámci transakce rozlišit toto pořadí změn. Výstup zobrazený na obrázku 11 je záměrně seřazen dle sloupce ID a AUD_SEQNO vzestupně. Zobrazený řádek číslo 6, tedy řádek s nejvyšším sekvenčním číslem pro zdrojový záznam s ID = 2 tedy indikuje výsledný stav daného záznamu ve zdrojové tabulce.

Tento stav lze ověřit na obrázku číslo 10, který zachycuje stav zdrojové tabulky po provedení příkazů. Zde je také zobrazen záznam s ID = 2 a jak je z obrázku patrné, tento záznam se shoduje se záznamem na obrázku číslo 11 na řádku 6.

Pomocí těchto technických polí v rámci mapování LiveAudit je možné vyvodit cílový stav zdrojové tabulky. S využitím druhého pole, které obsahuje časovou známku, lze dokonce vyvodit stav zdrojové tabulky k jakémukoli minulému bodu v čase, případně je možné rekonstruovat události a změny na zdrojovém systému. Pro příklad lze použít již zmíněný záznam s ID = 2, pro který poslední změna proběhla 19. 11. 2016 v 19:45:42, kdy byl Jiří Cipísek přeřazen z oddělení A00 do oddělení A10.

6.6.3.3 Mapování Adaptive Apply

Poslední typ mapování, který byl vyzkoušen v testovacím prostředí, byl typ Adaptive Apply. Na straně zdrojového systému byla přidána tabulka „zamestnanci_AA“. Tato tabulka byla také vytvořena v cílovém systému Netezza. Bylo vytvořeno mapování těchto dvou tabulek, které bylo přidáno do subscription.

Typ mapování Adaptive Apply je velmi podobný typu Standard, který již byl popsán. Rozdíl tohoto mapování spočívá v tom, že umožňuje přenášet změny ze zdrojového systému do cílového i když zdrojové a cílové tabulky nejsou synchronizované (tedy obsahují rozdílná data). Toto chování bylo vyzkoušené v testovacím prostředí, kdy byl zapnut mirroring bez prvotního refreshe.

V momentě, kdy byla provedena změna určitého záznamu na zdrojovém systému, který ještě v cílovém neexistoval, bylo na cílovém systému provedeno vložení výsledného řádku.

Tento typ mapování tedy umožňuje přenášet změny do cílového systému a případně kompenzovat určité nekonzistence v datech ve zdrojovém a cílovém systému. Tím lze

uživateli nabídnout aktuálně měněná data v cílovém systému bez nutnosti provádět prvotní synchronizaci pomocí refreshe.

6.7 Zhodnocení konceptu

Tato subkapitola obsahuje výsledky a zjištění, které byly shromážděny během testování v testovacím prostředí. Celý koncept je také zhodnocen s využitím výsledků z testovacího prostředí a na základě osobní zkušenosti autora s tímto konceptem při jeho použití v reálných produkčních podmínkách.

Hlavní přínosy tohoto konceptu byly potvrzeny. S jeho využitím je možné průběžně získávat změny v datech ze zdrojových systémů a přenášet je do cílového systému bez značného vytížení zdrojové databáze. Při použití takzvaného zrcadlení (mirroringu) není zdrojová databáze vytěžována dalšími dotazy, ale změny jsou vyčítány z databázových logů zdrojového systému.

Z testu také vyplynulo, že při zastavení zrcadlení (mirroringu v rámci dané subscription) a jeho opětovném spuštění není potřeba vždy provádět takzvaný refresh, tedy celkovou synchronizaci dat mezi zdrojovým a cílovým systémem. Pokud jsou k dispozici logy, které obsahují všechna data od doby přerušení, CDC agent umí navázat na toto místo a všechny změny dočíst zpětně bez nutnosti použití klasického dotazu do zdrojové databáze.

Tyto změny ze zdrojových systémů je možné zpracovávat v téměř reálném čase pomocí JMS Messages nebo ETL nástroje, kterým je InfoSphere DataStage. Jak také bylo vyzkoušeno v testovacím prostředí, tyto změny je možné průběžně nahrávat přímo do cílové databáze tvořící datový sklad, v kterém může probíhat další komplexní zpracování těchto dat nebo tato data mohou být k dispozici uživatelům a jejich nástrojům.

V rámci testování byly vyzkoušeny všechny tři hlavní typy mapování a to Mapování Standard, Mapování LiveAudit a Mapování Adaptive Apply.

6.7.1 Možnosti využití mapování Standard

Mapování Standard přenáší změny ze zdrojových systémů tak, že se na cílovém systému propisují zcela identicky. Z pohledu uživatele jsou tedy data dostupná stejně, jako kdyby

přistupoval přímo do zdrojového systému. To může být pro uživatele velmi přínosné zejména v případech, kdy s daty potřebuje pracovat jiným způsobem, než pro jaký je uzpůsobený zdrojový systém. Například v momentě, kdy produkční data dobře zná a potřebuje nad nimi provést analyticky složité úlohy. Toto využití dat nemůže provést na zdrojovém systému, který pro toto použití není uzpůsoben a není možné ho pro výpočetně náročné analytické úlohy použít buď vůbec, nebo proto, aby nedošlo ke komplikacím v jeho běžném provozu. Pomocí mapování Standard je tedy možné relativně snadno data ze zdrojových systémů nabídnout koncovým uživatelům ve stejné formě jako jsou na zdrojových systémech a to s velmi malým zpožděním. Toto zpoždění lze určit a to zejména vzhledem k potřebám cílového systému. V rámci testovacího prostředí bylo toto zpoždění nastaveno na jednu minutu a změny ze zdrojového systému se skutečně v tomto časovém intervalu přenášely do cílového systému.

Z pohledu datového skladu lze toto mapování použít pro potřeby dočasného uložiště, kdy v určitých časových okamžicích by tato data mohla být dále použita pro další zpracování a pro integraci dat z dalších zdrojů. Výhodou tohoto použití je, že data jsou k dispozici kontinuálně a není potřeba spoléhat například na časově náročnější celkový export dat. Toto mapování by také mohlo být použito jako základ pro budování operativního uložiště dat, kdy bylo možné využít právě velmi malého zpoždění, které je mezi daty v cílovém a zdrojovém systému.

6.7.2 Možnosti využití mapování LiveAudit

Mapování LiveAudit přenáší změny ze zdrojových systémů tak, že každá změna generuje nový záznam v cílovém systému a s použitím technických polí je možné určit, o jakou změnu se jedná. Toto chování může být pro datový sklad užitečné z více důvodů. Zaprvé může výrazně urychlit zpracování dat do historické kolekce dat, protože z informací v cílovém systému lze jednoznačně určit, jaké záznamy byly na zdrojovém systému změněny a také to, jak tyto záznamy byly změněny. Pomocí těchto informací je možné sestavit na cílovém systému rozdíl mezi stavem na zdrojovém systému a stavem v historizované vrstvě datového skladu. S využitím tohoto rozdílu není potřeba provádět výpočetně náročné nalezení změn v datech pomocí porovnávání dat získaných ze zdrojového systému a stavem v historizované vrstvě datového skladu. Toto mapování tedy

může sloužit jako zdroj dat pro dočasné uložiště a umožnit tak další efektivní zpracování dat.

Druhou výhodou je možnost data zpracovávat k určitým časovým bodům. Například pro potřeby reportingu, kdy by bylo potřeba dodat výsledky reportu ke specifické události. Datové sklady již ze své povahy pracují s historickou kolekcí dat. Z různých důvodů jsou tyto kolekce často používány s granularitou jednoho dne. Datový sklad je poté schopen odpovídat na dotazy, jaký stav dat byl ke koncům jednotlivých dnů. Pomocí využití kontinuálního přenosu dat ze zdrojových systémů může datový sklad pracovat s daleko jemnější granularitou historické kolekce dat, než je jeden den. V případě nutnosti může data také dopočítat k přesnému bodu v čase. V případě testovacího prostředí byla data přenášena s přesností na sekundy. S využitím této informace i při použití denní historické kolekce dat je datový sklad v případě potřeby schopný dopočítat stav zdrojového systému s přesností na jednu sekundu. Toho lze využít, například pokud je potřeba generovat reporty v pravidelných intervalech pro daný stav k určitému času během dne. Tím lze dosáhnout lepší porovnatelnosti stejných reportů generovaných v pravidelných intervalech, protože je zajištěno, že daný report obsahuje data skutečně k danému okamžiku.

Při uchování těchto změn v cílovém systému je možné zvažovat jeho použití jako auditního systému pro zdrojové systémy. Jak již bylo zmíněno v předchozím odstavci, pokud tato data zůstanou dostupná pro datový sklad, je možné dopočítat stav zdrojového systému k danému bodu v čase. V případě zjištění nesrovnalostí je možné data zpětně dopočítat a nalézt tak případnou chybu, která například mohla nastat v transformacích v dalších vrstvách datového skladu. V případě auditu je také možné doložit, na základě jakých dat report vznikl a jaký stav kdy byl na zdrojovém systému. Toto řešení by mohlo být zajímavé s využitím platformy pro Big Data jakou je například Hadoop, kdy by tato starší auditní data byla po určité době přesunuta do tohoto levnějšího uložiště s možností dotazování a případné další práce s těmito nezpracovanými daty.

Vzhledem k těmto možným výhodám, které přináší toto mapování, je kapitola 7. Historizace zaměřena zejména právě na možnosti dalšího zpracování takto získaných dat ze zdrojových systémů.

6.7.3 Možnosti využití mapování Adaptive Apply

Posledním testovaným mapováním byl typ Adaptive Apply. Z pohledu uživatele je tento typ mapování stejný jako typ mapování Standard. Výhodou tohoto mapování je to, že umožňuje získávat změny ze zdrojového systému i v případech, kdy zdrojové a cílové tabulky neobsahují stejné datové základny. V tomto momentě je tento typ mapování schopný překlenout určitou nekonzistenci a například vhodně měnit typ operací na cílovém systému. Tento typ mapování je vhodný v případech, kdy je potřeba uživatelům v cílovém systému nabídnout aktuálně měněná data ve zdrojových systémech, ale není možné provést prvotní synchronizaci zdrojového systému a cílového systému.

6.7.4 Zjištěná rizika a omezení

Určitým rizikem a omezením jsou případné změny ve strukturách tabulek na zdrojovém systému. V takovém případě může dojít k zastavení přenášení změn ze zdrojového systému do cílového a je nutné provést znovu mapování dat.

Použitý nástroj IBM InfoSphere Change Data Capture obsahuje funkcionalitu, která dokáže do určité míry podchytit i změny ve strukturách tabulek na zdrojovém systému a správně je propagovat do cílového systému. Tato funkcionalita je aktuálně podporována pro zdrojový replikační engine pro databázi Oracle a databázi DB2 for LUW (IBM, 2016g). V rámci testovacího prostředí nebyla tato funkcionalita důkladněji vyzkoušena.

Ať s využitím této funkcionality či bez jejího využití je v případě použití tohoto konceptu určitě vhodné zvážit začlenění této technologie do standardního postupu nasazování změn na produkční prostředí. Tedy v případě provádění změn na zdrojovém systému mít k dispozici možnost otestovat dopad těchto změn do technologie CDC ještě před jejich nasazením tak, aby byl dostatek času na odhalení a zamezení případných nežádoucích dopadů.

7. Historizace

Tato kapitola se zabývá historizací, tedy tím, jak je možné uchovávat data a informaci o tom, kdy bylo možné tato data považovat za platná nebo zda tato data jsou stále platná. Toto téma je velmi široké a lze se setkat s různými přístupy k řešení této problematiky. Cílem této kapitoly není zmapování různých přístupů k této problematice a ani nalezení optimálního řešení použitelného v různých případech v datovém skladu. Tato kapitola se zabývá možností historizace dat po použití konceptu CDC. Zejména se jedná o situaci z předešlé kapitoly, kdy z primárních systémů jsou získávána pouze data, která jsou změněna a informace o těchto změnách.

Cílem této kapitoly je poskytnout stručný úvod do dané problematiky a implementovat použitelné řešení v testovacím prostředí, které by uživatelům umožnilo práci s daty ze zdrojových systémů i v případě, kdy jsou ze zdrojových systémů získávána pouze změněná data a informace o těchto změnách. Dalším cílem je vyzkoušet toto řešení v testovacím prostředí a na základě výsledků z testovacího prostředí a poznatků z praxe vyvodit možnosti použití v datovém skladu.

7.1 Úvod do historizace

Jak již bylo zmíněno v úvodu této kapitoly, jedná se o velmi široké téma. Pro základní přehled této problematiky lze zmínit pojetí od Ralpha Kimballa (Kimball et. al., 2013), který se této problematice věnuje v souvislosti s pomalu měnícími se dimenzemi (Slowly Changing Dimensions, zkráceně také SCDs). Ralph Kimball v této publikací popisuje možnosti řešení této problematiky, které dělí do osmi různých typů (typ nula až typ sedm). Tyto možnosti řešení uchování historie dat jsou ve zmíněné publikaci popsány zejména v kontextu použití datových tržišť případně celé architektury nezávislých datových tržišť, kdy je předpoklad použití dimenzionálního modelu, tedy struktury faktových tabulek a souvisejících tabulek obsahující dimenze.

Jak je zřejmé z předchozí kapitoly věnující se rozšiřujícímu konceptu CDC, data získaná tímto způsobem nejsou ve struktuře dimenzionálního modelu, ale ve struktuře stejné nebo

velmi podobné zdrojovému systému[1]. Z tohoto důvodu použití těchto technik jako celku není v tomto konkrétním případě vhodné. Jako vhodnější se jeví použití konceptu takzvaných Temporálních dat. Tento koncept a jeho práce s ním je obsáhle popsána například v (Johnston a Weis, 2010). Tento koncept není závislý na zvoleném modelu dat a funguje na principu, kdy každý záznam obsahuje i informaci o své platnosti. Z tohoto důvodu ho lze použít i pro historizaci dat po použití konceptu CDC.

7.2 Možnosti historizace po použití konceptu CDC

V šesté kapitole, věnované konceptu Change Data Capture (CDC), byly vyzkoušeny tři základní typy mapování, které umožňují přenos dat ze zdrojových systémů do cílového systému. Jak z této kapitoly vyplývá mapování Standard a mapování Adaptive Apply v sobě neobsahují informaci o časové dimenzi přenášených dat a při použití těchto mapování data dostupná v cílovém systému odráží stav dat ve zdrojovém systému.

Za předpokladu, kdy je potřeba v cílovém systému udržovat historickou kolekci dat (typickým příkladem takového cílového systému je datový sklad), je potřeba data vhodně transformovat. V takovém případě lze výstup z CDC považovat za první vrstvu datového skladu, kterou je možné označit jako první nultou vrstvu L0 (zkratka z anglického slova layer). Z této vrstvy je potřeba data transformovat do další vrstvy L1, která již obsahuje historickou kolekci dat.

Při použití dvou výše uvedených mapování se nabízí otázka jak tuto transformaci efektivně provést. Pokud zanedbáme přenosové a technologické zpoždění mezi zdrojovým systémem a vrstvou L0, tato vrstva obsahuje identický stav dat jako je na zdrojovém systému. Naopak vrstva L1 obsahuje poslední známý stav zdrojových systémů provedený v rámci poslední transformace. Tím, že nemáme k dispozici informaci, která data byla změněna v rámci vrstvy L0, musí se během této transformace L0 do L1 provést porovnání dat L0 vůči poslednímu dostupnému stavu dat ve vrstvě L1.

[1] Jak bylo zmíněno v kapitole 6.3 IBM InfoSphere Change Data Capture, zvolený nástroj CDC je schopen získaná data směřovat přímo do nástroje ETL IBM DataStage. Pomocí této kombinace nástrojů CDC s ETL je možné získaná data přímo transformovat do vhodné struktury dimenzionálního modelu a takto transformovaná data nahrát do cílového systému. V jistých případech konkrétních datových skladů může být toto použití vhodné. Tato práce se tímto použitím nástroje CDC a ETL nezabývá, ale věnuje se použitím CDC nástroje tak, jak bylo popsáno v šesté kapitole.

Toto porovnání nutné pro identifikaci změn ve zdrojovém systému vůči historické kolekci ve vrstvě L1 je výkonově poměrně náročné a je spojené s dalšími možnými nevýhodami, které jsou podobné jako v případě použití celkového exportu dat ze zdrojových systémů, který byl popsán v kapitole 6.2.3 Export dat. Jedná se například o nutnost vymezení časového okna přepočtu tak, aby změny byly do vrstvy L1 promítány ve stejných časových intervalech (například jednoho dne).

Třetí vyzkoušené mapování v kapitole šesté bylo mapování LiveAudit. Toto mapování oproti předchozím dvěma zmíněným obsahuje časovou dimenzi přenášených dat a to ve formě, kdy každý přenesený záznam obsahuje časovou značku určující čas změny na zdrojovém systému, ke které se daný záznam váže. Oproti předchozím dvěma mapováním přenáší do cílového systému pouze změny ze zdrojového systému. Při použití tohoto mapování nelze jednoduše zjistit stav zdrojového systému, ale tento stav se musí ze získaných změn dopočítat.

V kapitole šesté jsou také zmíněny možné výhody použití tohoto mapování LiveAudit, jako je zaznamenávání a ukládání pouze změn v cílovém systému, možnost zpracování dat k různým časovým bodům (bez nutnosti pevného časového okna zpracování) a možnosti použití tohoto mapování jako určité auditní vrstvy.

Z těchto všech výše popsaných důvodů bylo mapování LiveAudit vybráno pro další použití v této kapitole, kdy na základě dat dostupných z CDC s využitím tohoto mapování budou data dále zpracována do historické kolekce dat.

7.3 Důvody pro historizaci po použití CDC

Jak z předešlé kapitoly vyplývá, hlavním důvodem historizace je vytvoření historické kolekce dat z vrstvy, která je výstupem CDC. V případě prvních dvou mapování Standard a Adaptive Apply je pro vytvoření historické kolekce takovéto zpracování dat nutností, protože po získání těchto dat ze zdrojových systémů nemají tato data žádnou časovou dimenzi. Informace o časové dimenzi vznikne teprve v čase jejich zpracování, kdy tento čas tvoří tuto dimenzi, protože právě v tento čas data vyjadřují stav na zdrojových systémech. Toto chování tedy vynucuje zpracování v pravidelných časových oknech například každý den v určený noční čas, kdy jsou data přímo zahistorizována do další

vrstvy, anebo je alespoň vytvořen jejich snímek, který se uchová a je poté zdrojem pro zpracování.

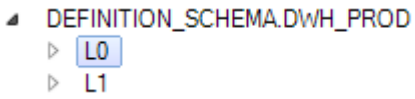
Mapování LiveAudit již určitou datovou dimenzi obsahuje a to ve formě pole AUD_TIME, které obsahuje časovou známku změny na zdrojovém systému. Oproti předchozím dvěma mapováním toto mapování na výstupu nezobrazuje stav zdrojového systému, ale pouze veškeré změny, které ve zdrojovém systému nastaly. Práce s takovými daty by byla velmi složitá a výpočetně náročná, proto je vhodné i takto získaná data dále zpracovat do další vrstvy. Hlavním důvodem pro toto zpracování je nabídnout data v přijatelné formě, která odráží stav na zdrojovém systému a není nutné tento stav složitě dopočítávat. Dalším důvodem je vytvoření historické kolekce dat, která umožní efektivní práci s těmito daty bez nutnosti historii konkrétních záznamů zpětně dopočítávat dle časových známek a provedených změn.

7.4 Nasazení v testovacím prostředí

Pro nasazení a vyzkoušení rozšiřujícího konceptu historizace v testovacím prostředí byl zvolen následující postup. Výstup CDC s mapováním LiveAudit bude tvořit první vrstvu L0. Z této vrstvy budou následně data zpracována do vrstvy L1. Vrstva L0 tedy obsahuje změny provedené na zdrojovém systému, kdy každá změna obsahuje typ změny, časovou značku a sekvenční číslo změny (tak jak bylo popsáno v kapitole šesté, věnované CDC). Vrstva L1 bude tvořit historickou kolekci dat ze zdrojových systémů. Tato vrstva tedy nebude obsahovat pouze změny, ale bude obsahovat vždy celkový stav zdrojového systému. Z pohledu uživatelů nebo dalšího zpracování bude možné s vrstvou L1 pracovat jako s daty zdrojových systémů. Tím, že vrstva L1 bude obsahovat historickou kolekci dat, bude možné si zvolit, k jakému období je potřeba s daty pracovat. Toto zvolené období poté reprezentuje stav zdrojového systému právě k danému období. Pro vytvoření historické kolekce dat bude použit koncept Temporálních dat, kdy každý záznam bude obsahovat informaci o své platnosti, která vyjadřuje, kdy daný záznam existoval s danými atributy ve zdrojovém systému.

7.4.1 Implementace v databázovém systému Netezza

Obě dvě vrstvy L0 a L1 budou umístěné v databázovém systému Netezza. Pro tento účel bude v Netezze vytvořena databáze DWH_PROD, ve kterém budou vytvořena dvě schémata L0 a L1. Tyto schémata reprezentují dané vrstvy L0 a L1. V těchto schématech budou následně umístěny relační tabulky tvořící dané vrstvy. Pomocí použití schémat budou relační tabulky daných vrstev od sebe logicky odděleny. Tyto popsané databázové struktury jsou přehledně zobrazeny na obrázku číslo 12.



Obr. 12 – Databáze DWH_PROD a dvě schémata L0 a L1, reprezentující dané vrstvy (zdroj: autor)

V těchto schématech budou umístěné relační tabulky potřebné pro vyzkoušení tohoto konceptu. Data použitá pro naplnění vrstvy L0 vychází z kapitoly 6.6.3.2 Mapování LiveAudit. Pro vytvoření obrázků použitých v této práci a následné vysvětlení postupu a výsledků byla vytvořena tabulka ZAMESTNANCI, která obsahovala data záměrně upravená tak, aby z ní byly patrné použité principy a aby výstupy byly snadno zachytitelné v této práci.

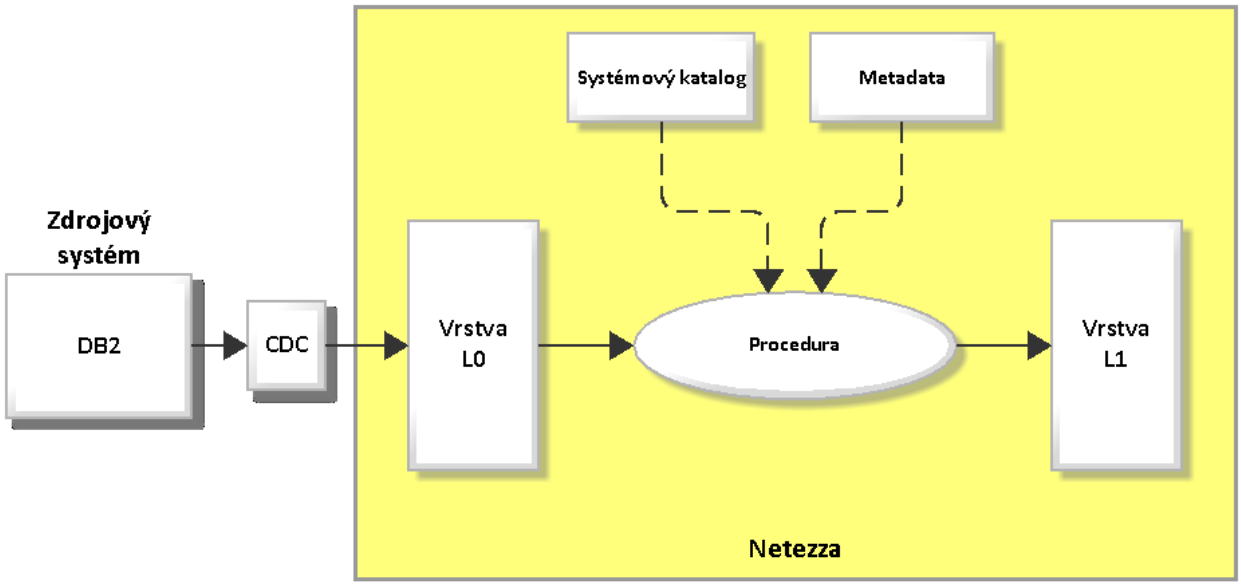
Pro vrstvu L1 je potřeba dále určit granularitu historické kolekce, tedy do jaké úrovně podrobnosti bude možné sledovat změny na zdrojovém systému. V datových skladech se obvykle volí granularita jednoho dne, kdy historická kolekce zachycuje stavy zdrojových systémů ke koncům jednotlivých dnů. Například ve finančním odvětví se často zvolí granularita jednoho účetního dne, kdy ohraničení jednotlivých dnů je dáno účetní uzávěrkou mezi jednotlivými dny.

V některých případech, nemusí být tato volba ideální, ale i přesto je zvolena a to z důvodů technologického omezení. V momentě, kdy architektura řešení je například závislá na nočních exportech dat ze zdrojových systémů (snímek stavu zdrojového systému), nemá datový sklad ani jinou možnost než zvolit granularitu jednoho dne k času, ke kterému mu jsou data dodána.

Pro vyzkoušení v testovacím prostředí byla zvolena granularita jednoho kalendářního dne, kdy daný den začíná v čase 00:00:00 a končí v čase 23:59:59 (oboustranně uzavřený interval). Jelikož ve vrstvě L0 máme informaci o změnách s přesností na jednu sekundu, je možné tuto granularitu volit dle konkrétních business potřeb, například s granularitou po osmi hodinách, které by například ve výrobním podniku s nepřetržitým provozem kopírovaly tři osmihodinové směny.

7.4.2 Zpracování dat mezi vrstvami

Pro zpracování dat mezi vrstvami byly využity informace dostupné ze samotné vrstvy L0, které byly vytvořené v rámci přenosu CDC. Jedná se o již dříve popsaná technická pole, která obsahují časovou známku změny na zdrojovém systému, typ změny, která nastala na zdrojovém systému a sekvenční číslo změny. Pro úvodní naplnění vrstvy L0 a následné zachytávání změn byla použita metoda přenosu dat Refresh before Mirror. Pro samotné zpracování byly dále potřeba metadata o strukturách tabulek. Tato metadata byla získána ze systémového katalogu databázového systému Netezza. Poslední nutnou informací ke zpracování dat je informace o granularitě historické kolekce. Tato informace byla získána z metadatových tabulek, které byly ručně plněny. V praxi jsou tyto informace často poskytovány systémem, který řídí zpracování dat v datovém skladu. Samotné zpracování dat bylo poté provedeno procedurou. Tato procedura byla záměrně napsána tak, aby optimálně využila výkon masivně paralelního zpracování, které databázový systém Netezza nabízí. Při použití v reálných podmínkách datového skladu je samozřejmě možné použít pro samotné zpracování jiné řešení a to v závislosti na požadavcích a použité technologii. Celé řešení použité v testovacím prostředí je přehledně zobrazeno na následujícím obrázku číslo 13. Nepřerušované šipky znázorňují tok dat od zdrojového systému, přes CDC a vrstvu L0 až do cílové vrstvy L1. Přerušované šipky znázorňují použití metadat pro samotný výpočet.



Obr. 13 – Zpracování dat mezi vrstvami L0 a L1 v databázovém systému Netezza (zdroj: autor)

7.5 Vyzkoušení konceptu v testovacím prostředí

Pro vyzkoušení tohoto konceptu byly využity tabulky a data získána v kapitole 6.6.3.2 Mapování LiveAudit. Tyto tabulky a data tvořily datovou základnu pro vrstvu L0. Následně byla vhodně naplněna metadata určující granularitu jednoho dne se začátkem v čase čase 00:00:00 a koncem v čase 23:59:59. Poté bylo pro jednotlivé dny, kdy probíhalo získávání dat ze zdrojových systémů pomocí CDC a mapování LiveAudit spuštěno samotné zpracování dat pomocí procedury.

7.5.1 Vrstva L0

Následující obrázek číslo 14 zobrazuje data tabulky ZAMESTNANCI ve vrstvě L0. Výstup zobrazený na obrázku je záměrně seřazen podle sloupce ID a AUD_SEQNO vzestupně.

Output | Result 1

Drag a column header here to group by that column.

	AUD_TYPE	AUD_TIME	AUD_SEQNO	ID	JMENO	ODDELENI	PLAT
1	PT	2016-11-19 19:45:36	1479585543261	1	Martin	A00	45000,00
2	PT	2016-11-19 19:45:36	1479585543262	2	Jan	A00	62000,00
3	UB	2016-11-20 16:33:48	1479585543374	2	Jan	A00	62000,00
4	UP	2016-11-20 16:33:48	1479585543375	2	Jan	A10	62000,00
5	UB	2016-11-21 11:57:12	1479585543715	2	Jan	A10	62000,00
6	UP	2016-11-21 11:57:12	1479585543716	2	Jan	A10	77000,00
7	PT	2016-11-19 19:45:36	1479585543263	3	Adam	B00	43000,00
8	UB	2016-11-21 09:05:55	1479585543688	3	Adam	B00	43000,00
9	UP	2016-11-21 09:05:55	1479585543689	3	Adam	B10	43000,00
10	UB	2016-11-21 14:18:01	1479585543771	3	Adam	B10	43000,00
11	UP	2016-11-21 14:18:01	1479585543772	3	Adam	B10	55000,00
12	PT	2016-11-19 19:45:36	1479585543264	4	Karel	A00	42000,00
13	DL	2016-11-20 17:24:49	1479585543399	4	Karel	A00	42000,00

Obr. 14 – Stav tabulky ZAMESTNANCI ve vrstvě L0 před zpracováním (počet řádků i sloupců je zkrácen) (zdroj: autor)

Na tomto obrázku číslo 14 jsou patrné změny ve zdrojovém systému tak, jak byly v čase provedeny pro jednotlivé záznamy. Interpretace jednotlivých hodnot technických polí byla již detailněji popsána v kapitole šesté, věnované konceptu CDC.

Jak je na tomto obrázku znázorněno, záznam s ID = 1, byl pouze 19. 11. 2016 do zdrojového systému vložen a následně již nebyl měněn. Oproti tomu záznam s ID = 2, byl vložen 19. 11. 2016 a poté byl dvakrát změněn. Dne 20. 11. 2016 v 16:33:48 byl proveden jeho UPDATE, kdy byla změněna hodnota v atributu oddělení (řádek s číslem 3 a 4) a příští den byl opět proveden UPDATE tohoto řádku, kdy byla změněna hodnota v atributu plat z 62 tisíc na 77 tisíc (řádek s číslem 5 a 6).

Při pohledu na záznam s ID = 3, je vidět, že měl obdobné změny jako záznam s ID = 2. Záznam byl prvně do systému vložen a poté mu byly změněny dva atributy a to oddělení a plat. Tyto obě změny ale proběhly v rámci jednoho dne.

Poslední záznam s ID = 4 znázorňuje jinou situaci. Tento záznam byl ve zdrojovém systému vytvořen 19. 11. 2016 a následující den byl pouze smazán, čímž na zdrojovém systému přestal existovat.

Jak je i z tohoto obrázku patrné a již bylo popsáno dříve, práce s takto získanými daty bez dalšího zpracování by byla velice náročná, protože je nutné pro každý záznam dohledávat jeho platnou hodnotu k požadovanému časovému bodu na základě provedených změn.

7.5.2 Zpracování dat a výstupy ve vrstvě L1

Data ve vrstvě L0 byla zpracována a transformována do podoby Temporálních dat ve vrstvě L1. Takto zpracovaná data jsou zobrazena na obrázku číslo 15. Tento obrázek obsahuje zpracovaná data tabulky ZAMESTNANCI z vrstvy L0, která byla zobrazena na obrázku číslo 14. Výstup zobrazený na obrázku je záměrně seřazen podle sloupce ID a VALID_FROM vzestupně.

Output | Result 1

Drag a column header here to group by that column.

	VALID_FROM	VALID_TO	ID	JMENO	ODDELENI	PLAT
1	2016-11-19	2999-12-31	1	Martin	A00	45000,00
2	2016-11-19	2016-11-19	2	Jan	A00	62000,00
3	2016-11-20	2016-11-20	2	Jan	A10	62000,00
4	2016-11-21	2999-12-31	2	Jan	A10	77000,00
5	2016-11-19	2016-11-20	3	Adam	B00	43000,00
6	2016-11-21	2999-12-31	3	Adam	B10	55000,00
7	2016-11-19	2016-11-19	4	Karel	A00	42000,00

Obr. 15 – Stav tabulky ZAMESTNANCI ve vrstvě L1 po zpracování (počet řádků i sloupců je zkrácen) (zdroj: autor)

Jak je zobrazeno na obrázku, ve vrstvě L1 obsahuje tabulka dvě technická pole VALID_FROM a VALID_TO. Tyto pole udávají, od kdy do kdy daný záznam obsahoval jaké hodnoty na zdrojovém systému, respektive kdy existoval ve zdrojovém systému. Pole VALID_FROM obsahuje datum, kdy daný záznam začal existovat na zdrojovém systému s danými hodnotami. Pole VALID_TO obsahuje datum, do kdy daný záznam existoval ve zdrojovém systému s danými hodnotami. Pokud podle poslední známé informace ze zpracování daný záznam ještě existuje ve zdrojovém systému, pak toto pole obsahuje hodnotu 2999-12-31.

S takto zpracovanými daty je již možné efektivně pracovat. Například pro získání aktuálního stavu dat ve zdrojovém systému, známého k poslednímu zpracování, stačí použít relační podmínku WHERE VALID_TO = date '2999-12-31'. Pro získání řezu historickou kolekcí tak, aby reprezentovala stav zdrojových systémů k určitému datu, lze použít například podmínku WHERE date '2016-11-20' BETWEEN VALID_FROM and VALID_TO. Tato podmínka provede řez historickou kolekci k 20. 11. 2016, čímž zobrazí stav, který byl ve zdrojové tabulce ke konci tohoto dne.

Jak je možné dále efektivně pracovat s temporálními daty lze nalézt v mnoha publikacích a odborných článcích například se jedná o již zmíněnou publikaci (Johnston a Weis, 2010), nebo o příspěvek z konference DEXA (Zhou et al 2006), který se věnuje výkonově efektivnímu dotazování nad těmito daty.

Z obrázku číslo 15 je také patrné, že zpracování dat zobrazených na obrázku 14 se provedlo úspěšně. Záznam s ID = 1 má v technických polích nastaveno, že začal existovat na zdrojovém systému 19. 11. 2016 a stále s danými hodnotami existuje (pole VALID_TO obsahuje hodnotu '2999-12-31').

Záznam s ID = 2 je správně uveden v historické kolekci třikrát (řádek číslo 2, 3 a 4). Jednou pro jeho prvotní vložení na zdrojovém systému dne 19. 11. 2016, poté jeho změnu dne 20. 11. 2016, kdy byl zaměstnanec Jan přeřazen z oddělení A00 do oddělení A10. Třetí výskyt záznamu je na řádku číslo 4, který zaznamenává změnu platu z 62 tisíc na 77 tisíc, která proběhla dne 21. 11. 2016 a dle posledního zpracování je stále aktuální.

Na tomto příkladu je patrné i chování primárního klíče. Ve zdrojovém systému v uváděné tabulce ZAMESTNANCI je primární klíč atribut ID. Ve vrstvě L1 se primární klíč stává složeným primárním klíčem, kdy primární klíč je definován primárním klíčem zdrojového systému a jedním technickým polem. V případě testovacího prostředí bylo zvoleno pole VALID_FROM. I když tedy historická kolekce vytvořená pomocí temporálních dat na první pohled strukturou připomíná Kimbalův druhý typ řešení pomalu měnících se dimenzí, je zde velký rozdíl s prací s primárními klíči, kdy typ dvě typicky používá umělý unikátní klíč.

Záznam s ID = 3, na kterém byly provedeny ve zdrojovém systému stejné změny jako se záznamem s ID = 2 (změna oddělení a platu) je uveden v historické kolekci pouze dvakrát. Je to z toho důvodu, že změny byly provedeny v rámci stejného dne a zvolená granularita

pro vytváření historické kolekce byla zvolena jako jednodenní. Proto ve vrstvě L1 je zachycen pouze výsledný stav (na obrázku číslo 15 řádek číslo 6). Z dat v L0 je přitom zřejmé (obrázek číslo 14), že změny proběhly postupně a to prvně změna oddělení v 9:05:55 a teprve poté změna platu v 14:18:01.

Na tomto příkladu je znázorněna důležitost volby granularity historické kolekce. V případě, kdy by byla například granuralita zvolena jako jednohodinová, byla by i tato jednotlivá změna ve vrstvě L1 zachycena. Při použití v reálném prostředí je tedy potřeba zvážit potřeby a požadavky na granularitu vůči výkonovým a objemovým dopadům, které s touto volbou souvisí.

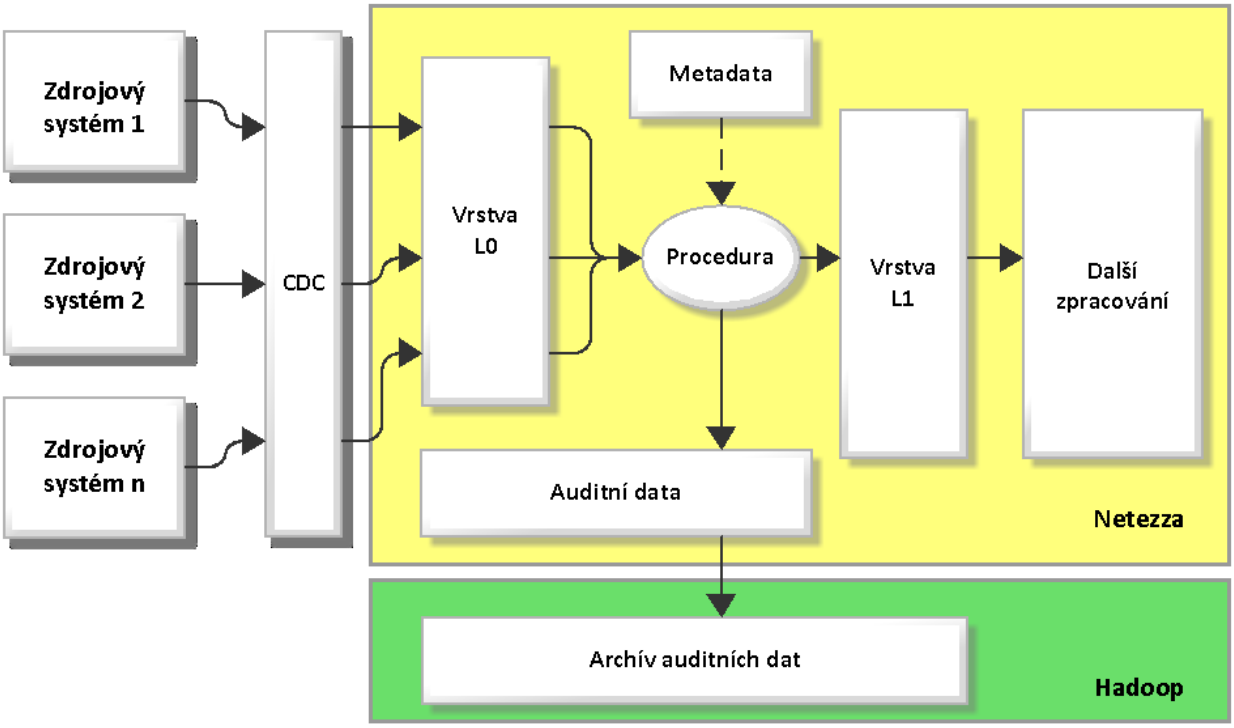
V souvislosti se zachycením změn na zdrojových systémech, které se z důvodů volby hrubší granularity nezpracovaly do následné vrstvy, souvisí i možnost použití výstupu z CDC jako auditní vrstvy, tak jak již bylo popsáno dříve. Jednalo by se o vrstvu, v které by byly dohledatelné veškeré změny, které se uskutečnily na zdrojovém systému. Po zpracování dat ve vrstvě L0 se tedy nabízí přesunutí těchto dat do jiného umístění, například do levnějšího uložiště, které by v případě potřeby umožňovalo dohledání veškerých změn.

Poslední záznam, který byl zpracován do vrstvy L1, je záznam s ID = 4. Tento záznam byl dne 19. 11. 2016 ve zdrojovém systému založen a během následujícího dne byl ze systému odstraněn. Na konci dne 20. 11. 2016 tento záznam již ve zdrojovém systému neexistoval a proto ve vrstvě L1 technická pole udávají, že existoval pouze 19. 11. 2016.

7.5.3 Vlastnosti a možnosti vrstvy L1

Takto vytvořená vrstva L1 historizuje stavy dat na zdrojových systémech, přičemž se jedná pouze o technickou historizaci, založenou na existenci a hodnotách daných záznamů ve zdrojových systémech. Pokud data obsahují dimenzi času (například data splatností faktur, časy pohybů na účtech apod.) tato dimenze je přenesená také ve formě relací či atributů, tak jak je navržen zdrojový systém. Pro větší efektivitu práce s takovými daty pro analytické účely je vhodné další zpracování, například pomocí transformace do vhodných struktur.

Výhodou této technické historizace je její univerzálnost. Pro její použití jsou potřeba technická pole ve vrstvě L0, jejichž existenci garantuje CDC. Dále jsou potřeba metadata o databázových strukturách, které je možné získat z databázového katalogu. Poslední nutnou informací jsou data o granularitě historické kolekce, které mohou být plněny systémem pro řízení datového skladu. Všechny informace nutné pro transformaci dat z vrstvy L0 do vrstvy L1 je tedy možné získat automaticky, bez nutnosti dalších analýz či ručních zásahů. Vzhledem k této vlastnosti by mělo být možné použít tento koncept jako univerzální řešení pro různé zdrojové systémy. Toto řešení je přehledně znázorněno na obrázku číslo 16.



Obr. 16 – Řešení zpracování dat ze zdrojových systémů (zdroj: autor)

Na obrázku číslo 16 je znázorněno možné zpracování dat z různorodých zdrojových systémů do vrstvy L1, kdy je použita procedura vyzkoušená v předchozí subkapitole, která na základě metadat univerzálně (bez ohledu na zdrojový systém) vytváří historickou kolekci ve vrstvě L1. Jak je znázorněno na obrázku, z této vrstvy L1 je poté možné další zpracování dat například pomocí klasických transformačních nástrojů, které data konsolidují a transformují do vhodného schématu zvoleného pro datový sklad. Pro usnadnění tohoto dalšího zpracování je možné mezi vrstvu L1 a vrstvu dalšího zpracování vložit další mezivrstvu, která pouze zprostředkuje patřičný řez historickou kolekcí pro

účely zpracování. Tuto mezivrstvu je možné také tvořit automaticky například pomocí generovaných databázových pohledů.

Na obrázku je také znázorněno přesunutí dat z vrstvy L0 do samostatného úložného prostoru nazvaného Auditní data. Do tohoto úložného prostoru jsou přesunuta všechna data, která již byla z vrstvy L0 zpracována do vrstvy L1. Při takovémto použití lze vrstvu L0 označit jako klasické dočasné uložiště dat (DSA), do kterého data kontinuálně přibývají ze zdrojových systémů a po zpracování do vrstvy L1 jsou tyto data transparentně přesunuty do samostatného uložiště.

Na obrázku je znázorněno, že toto uložiště je umístěné v databázovém systému Netezza. Dle požadavků a provozních potřeb konkrétního řešení při praktickém použití mohou data v tomto umístění určitý čas setrvat a posloužit například pro zpětné kontroly zpracování či dopočty stavů zdrojových tabulek k určitým časům. Po této době je možné tyto detailní data přesunout do jiného levnějšího uložiště, které by obsahovalo Archiv auditních dat. Tímto uložištěm může být například řešení postavené na technologii Hadoop, které je znázorněno na obrázku a které by umožnovalo i případné dotazování do těchto auditních dat.

Vrstva L1 v rámci tohoto řešení má následující výhody. Do vrstvy L1 je možné relativně jednoduše propagovat veškerá data dostupná ve zdrojových systémech a to z důvodů popsané univerzálnosti. Například při klasických implementacích v reálném prostředí datového skladu nemusí být dostatek času a pracovních kapacit k vytvoření transformačních úloh, které data propagují do dalších konsolidovaných vrstev datového skladu. Při tomto řešení jsou ve vrstvě L1 zachována všechna data a pokud jsou dovyvinuty další transformační úlohy plnící nové struktury v dalších vrstvách datového skladu, je možné tuto vrstvu L1, která obsahuje historickou kolekci dat, použít k dopočtení nových struktur a to včetně historie.

Tato vrstva také z velké části splňuje definici datového jezera (Data Lake) od společnosti Gartner (Gartner, 2016), tedy že se jedná o data ve stejné nebo skoro stejné podobě jako jsou data ve zdrojových systémech, která mohou být použita velmi zkušenými analytiky k analýzám, které nemusejí být možné v datových tržištích nebo datových skladech právě z důvodů možné ztráty informací, které mohou nastat v rámci použitých datových transformacích a agregacích.

Vrstva L1 s využitím Auditních dat také umožňuje zpracování dat k určitému přesně stanovenému času, případně s využitím dat v L0 dovoluje provést přepočet a nabídnout tak uživatelům aktuální pohled na zdrojové systémy i přesto, že je zvolena granularita větší, například jeden den. Zpětnému zpracování dat k určitému stanovenému času se detailněji věnuje následující podkapitola.

Z dlouhodobějšího pohledu lze také určitou část dat z vrstvy L1 přesouvat do levnějšího uložiště dat, jakým je již zmiňovaný Hadoop. V technologii datového skladu lze například mít uloženou historickou kolekci dat s historií jednoho roku, s kterou se předpokládá intenzivnější práce. Starší data lze v pravidelných intervalech přesouvat do levnějšího uložiště obdobně, jako v případě Auditních dat vrstvy L0.

7.5.4 Zpracování L1 k určitému času

Jak již bylo v této práci zmíněno s využitím CDC mapování LiveAudit je možné data zpracovat k přesně určenému času. Tato možnost je zachována i v případě vytvořené vrstvy L1. Nabízejí se v zásadě dvě možné situace.

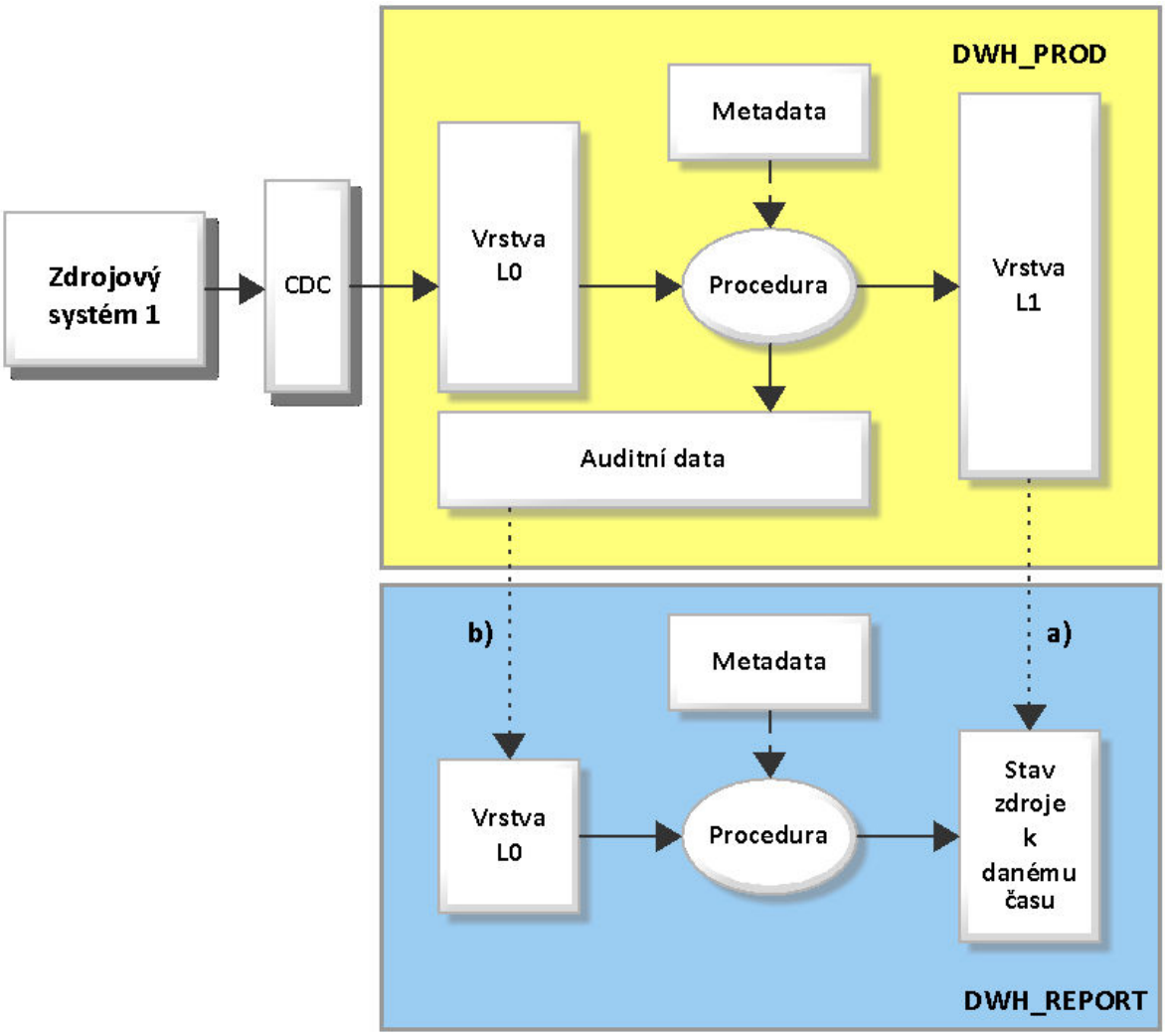
První z nich je stav, kdy je potřeba zpracovávat data, které jsou získávané k danému času. Jedná se například o generování reportů, které jsou založeny na datech získaných ze zdrojových systémů v pravidelných intervalech. Typickou situací může být report s metrikami výkonosti podniku s nepřetržitým provozem za daný den. Tento report je generován každý den na datech aktuálních k času 17:00. Tím je zachována porovnatelnost těchto reportů mezi jednotlivými dny. Nehledě na zvolenou granularitu je nutné mít ve vrstvě L1 zpracovaná data zdrojových systému k času 17:00 daného dne pro potřeby reportingu a teprve následně zpracovat zbytek dat podle zvolené granularity.

Řešení vytvořené v rámci vyzkoušení tohoto rozšiřujícího konceptu v testovacím prostředí tuto situaci dokáže jednoduše podchytit. I v rámci jednoho dne je možné provést řez vrstvou L0 podle časové známky a do vrstvy L1 zpracovat pouze tato data. Na těchto datech mohou být vygenerovány potřebné reporty a poté podle zvolené granularity je možné zpracovat zbývající data ve vrstvě L0 ke konci daného dne.

Druhou situací, která může nastat je stav, kdy v L1 již jsou zpracovaná data a je potřeba získat stav zdrojového systému k určitému času, který je starší než stav ve vrstvě L1 a

zároveň určený čas netvoří hranici v rámci zvolené granularity. Lze uvést příklad založený na datech použitých v rámci testovacího prostředí. Vrstva L1 je již zpracovaná k datu 21. 11. 2016. Existuje požadavek na dodávání reportu založeného na datech aktuálních k času 17:00 ze zdrojových systémů ke konci každého týdne, což znamená mít ve vrstvě L1 každou neděli dopočtená data z vrstvy L0 k času 17:00. Poté je zpracován požadovaný report, a následně po půlnoci proběhne zpracování zbývajících dat pro neděli (dle zvolené granularity jednoho dne). Dne 20. 11. 2016 (neděle) proběhlo zpracování dat k času 17:00, ale potřebný report z důvodů technické chyby vygenerován nebyl (mohlo se například jednot o chybu spojenou s novou verzí reportu). Tuto chybu se nepodařilo rychle opravit a vrstva L1 byla mezitím zpracována k 21. 11. 2016.

Řešením této situace je zpětné dopočítání stavu vrstvy L1 k určenému času, kdy toto zpracování je možné právě s využitím Auditních dat. Princip řešení zpětného dopočítání vrstvy L1 k určenému času je přehledně znázorněn na obrázku číslo 17.



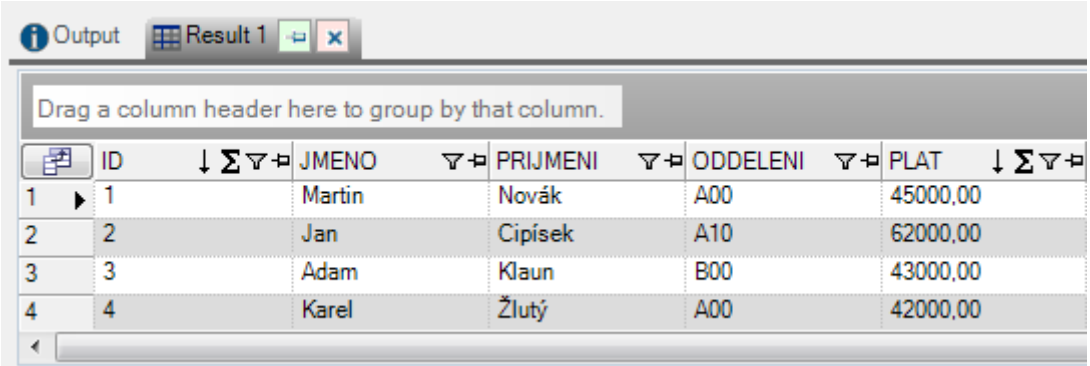
Obr. 17 – Princip zpětného zpracování dat k určitému času (zdroj: autor)

Princip zpětného zpracování dat k určitému času je založený na využití auditních dat. Jak již bylo popsáno, vrstva L1 je zpracována k 21. 11. 2016. Tento stav je v databázi DWH_PROD, která je na obrázku číslo 17 žlutě podbarvena. Pro potřeby zmíněného reportu je potřeba znát stav zdrojových systémů k neděli 20. 11. 2016 k času 17:00. Aby nedošlo k ovlivnění již napočítaných dat, je možné celé zpětné zpracování vyčlenit do samostatné databáze DWH_REPORT, která je na obrázku podbarvena modře.

Postup tohoto zpětného zpracování poté probíhá tak, že je potřeba prvně získat stav zdrojových systémů k předchozímu období, než pro které je potřeba dopočítat data. V konkrétním případě testovacího prostředí se jedná o předchozí den, než pro který je potřeba vygenerovat report, tedy je potřeba získat stav dat k 19. 11. 2016. Bez náročného zpětného dopočítávání dat z vrstvy L0 za mnoho předchozích období, je možné tento stav

získat z vrstvy L1 pomocí řezu historickou kolekcí. Na obrázku je tento krok znázorněný tečkovanou šipkou označenou jako a). Následně jsou obnovena auditní data, která obsahují veškeré změny, které proběhly na zdrojových systémech od konce předchozího období (19. 11. 2016) až ke konkrétnímu času pro který je potřeba vygenerovat report (neděle 20. 11. 2016 k času 17:00). Tento krok je znázorněn na obrázku tečkovanou šipkou označenou jako b). Po tomto kroku již stačí výkonově relativně efektivně promítnout provedené změny ve zdrojových systémech pomocí procedury do vrstvy pro zpětné zpracování L1, která je na obrázku popsána jako „Stav zdroje k danému času".

Podle tohoto popsaného principu bylo celé řešení vyzkoušeno v testovacím prostředí na uvedeném příkladu založeném na datech z vrstvy L0 (obrázek číslo 14), kdy tabulka ZAMESTNANCI byla zpětně zpracována k neděli 20. 11. 2016 k času 17:00. Výsledek tohoto zpracování je zobrazen na následujícím obrázku číslo 18, kde jsou znázorněna data v tabulce ZAMESTNANCI ve vrstvě pro zpětné zpracování L1. Z tohoto obrázku je patrné, že stále existuje záznam s ID = 4, protože jeho vymazání proběhlo až 20. 11. 2016 v čase 17:24:49. Z toho i vyplývá, že report založený na těchto datech správně uvede o jednoho zaměstnance více, než kdyby byl založen na datech existujících ve zdrojovém systému k 21. 11. 2016.


Output Result 1

Drag a column header here to group by that column.

	ID	JMENO	PRIJMENI	ODDELENI	PLAT
1	1	Martin	Novák	A00	45000,00
2	2	Jan	Cipísek	A10	62000,00
3	3	Adam	Klaun	B00	43000,00
4	4	Karel	Žlutý	A00	42000,00

Obr. 18 – Stav tabulky ZAMESTNANCI zpracované k času 17:00 (počet řádků i sloupců je zkrácen) (zdroj: autor)

Na tomto obrázku číslo 18 je také vidět, že změna ve zdrojovém systému v záznamu s ID = 2 se správně provedla, protože tato změna byla provedena již v čase 16:33:48, kdy zaměstnanec Jan byl přeřazen z jednoho oddělení do druhého. Také na tomto obrázku číslo 18 již nejsou zobrazená pole VALID_FROM a VALID_TO. Tyto pole je možné formálně uvést, nemají zde ale žádný význam. U takto zpracovaných dat se totiž již

nejedná o historickou kolekci dat, ale pouze o stav dat na zdrojovém systému k určitému času, proto je zbytečné tyto pole uvádět a proto ani nejsou zobrazeny na obrázku.

Celý tento proces zpětného zpracování je možné opět plně automatizovat a řídit podle dostupných metadat. V praxi by bylo možné použít popsaný princip u důležitých reportů, u kterých je nezbytné jejich vytvoření na základě stavu zdrojových systémů v určitý časový okamžik. Další možné využití může být u různých typů auditů, kdy toto řešení může poskytnout zpětné kontroly reportů a odpovědět na otázku, na základě jakých dat byl report vygenerován.

V praxi by toto řešení mohlo fungovat například tak, že uživatel by si přes webový portál zvolil, jaký report potřebuje vygenerovat na základě stavu dat zdrojových systémů k určitému zvolenému času. Poté by se mu na základě metadat automaticky připravily potřebné relační tabulky pro daný report zpracované k danému času v dočasné databázi (v uvedeném příkladu se jednalo o databázi DWH_REPORT) a celý report by se vygeneroval. Následně by bylo možné celou dočasnou databázi odstranit.

7.6 Zhodnocení konceptu

Tato subkapitola obsahuje stručný popis výsledků, které byly zjištěny během vyzkoušení tohoto rozšiřujícího konceptu v testovacím prostředí. Celý koncept je také na základě těchto výsledků a zkušeností autora zhodnocen a jsou zde stručně rekapitulovány jeho výhody, kterých by na základě testů mohlo být dosaženo při jeho použití v reálných podmínkách.

Tento rozšiřující koncept, kterému se věnovala tato kapitola, se zabýval historizací dat, zejména možností historizace po použití konceptu CDC, kdy ze zdrojových systémů jsou získávána pouze změněná data.

Celý koncept byl vyzkoušen v testovacím prostředí, kdy data, která byla výstupem CDC s využitím mapování LiveAudit, byla zpracována do podoby Temporálních dat. V rámci testování bylo také vyzkoušeno zpracování dat k určitému času, kdy výstupem byla data, které reprezentovaly stav dat na zdrojovém systému v určitý zvolený čas. Tento způsob zpracování dat k určitému času byl vyzkoušen jak v rámci daného dne, který měl být zpracován, tak v rámci takzvaného zpětného zpracování, kdy byl stav dopočten v dočasné

databázi k určenému minulému bodu v čase na základě auditních dat a dat v již zpracované vrstvě.

Na základě testů vyplynulo, že celý koncept je použitelný a umožňuje zpracování dat z výstupů CDC, kde jsou obsaženy pouze změny provedené na zdrojovém systému, do další vrstvy, která reprezentuje celkový obraz dat na zdrojovém systému včetně historie. Na základě tohoto použití bylo navrženo řešení zpracování dat ze zdrojových systémů, které bylo vysvětleno na obrázku číslo 16. Popsané principy navrženého řešení byly vyzkoušeny v testovacím prostředí včetně využití auditních dat.

Při vhodném použití tohoto rozšiřujícího konceptu například tak, jak je popsáno u navrženého řešení, lze dosáhnout následujících přínosů. Je možné dosáhnout značné univerzálnosti celého konceptu, kdy nehledě na zdrojový systém je možné vytvářet relativně jednoduše historické kolekce dat. Tyto historické kolekce dat poté mohou sloužit jako vstup pro následné transformační úlohy, které jsou obvykle pracnější a časově náročnější na vytvoření. Zvláště v případech, kdy tyto transformační úlohy jsou teprve vyvíjené, lze jim po jejich vytvoření a otestování nabídnout historická data a doplnit tak i zpětně nové datové struktury v dalších konsolidovaných vrstvách řešení datového skladu.

Další výhodou tohoto konceptu je možnost zpracování dat k určeným časům. Pokud je pro specifické potřeby nezbytné vycházet z dat platných k určeným časovým bodům například z důvodů porovnatelnosti reportů, nebo z potřeb sledovat vývojové trendy, toto řešení takového zpracování umožňuje a to i zpětně, kdy s využitím auditních dat lze dopočítat stav zdrojového systému k danému okamžiku.

Data zpracovaná pomocí tohoto konceptu lze také použít pro důkladné a/nebo velmi složité analýzy. Tyto data totiž obsahují historii dat zdrojových systémů a zároveň jsou ve stejné podobě jako data zdrojových systémů. Je zde tedy minimální riziko možné ztráty určité informace, které existuje v obvyklých transformačních a agregačních datových úlohách. Také je zde minimalizované riziko možných chyb, které mohou být omylem způsobené transformačními úlohami.

Při použití tohoto rozšiřujícího konceptu je potřeba zvážit četnost zpracování vzhledem k výpočetní náročnosti. I když v tomto případě nedochází k porovnání dvou plných snímků, tedy celkovému stavu dat ve zdrojových systémech a poslednímu dostupnému stavu dat v historické kolekci, ale jsou v tomto případě zpracovávané pouze provedené

změny na zdrojovém systému, určitá výpočetní náročnost na udržování historické kolekce zde existuje, zvláště pak v porovnání s řešeními, které historickou kolekci dat zdrojových systémů neudržují. V každém případě lze doporučit vhodně využívat optimalizace, které zvolená technologie pro zpracování dat nabízí. Například podle osobních zkušeností autora je možné u tohoto typu výpočetní úlohy na masivně paralelním systému Netezza dosáhnout pomocí vhodné distribuce dat a dalších pokročilých optimalizačních technik značné úspory výpočetního výkonu, které vede k výraznému urychlení celého zpracování.

Určitým rizikem, které je spojeno s tímto rozšiřujícím konceptem, je zejména v dlouhodobém horizontu nárůst objemu uchovávaných dat. Před zavedením tohoto konceptu je vhodné provést analýzy za účelem odhadnutí datového objemu historické kolekce a tempa jejího růstu. Jak bylo popsáno v této kapitole, řešením tohoto problému může být přesouvání starších dat z historické kolekce do jiného systému, který nabízí relativně levné uložení těchto dat. V případě potřeby je možné i historickou kolekci dat odmazávat a udržovat v ní například pouze poslední rok, tak aby bylo dosaženo optimálního poměru mezi cenou datového uložiště a přidanou hodnotou v ní obsažených dat.

8. Analytické funkce a datový sklad

Tento rozšiřující koncept se zabývá možnostmi použití analytických funkcí přímo v technologii datového skladu bez nutnosti vytvářet dedikovaný analytický server. Tento přístup nabízí určité výhody, které vyplývají z tohoto použití. Jedná se zejména o zamezení přesouvání dat mezi datovým skladem a analytickým serverem a o využití výpočetního výkonu technologie datového skladu. Při tomto použití lze říci, že se minimalizují náklady na dodatečnou úložnou kapacitu analytického serveru, náklady na výkonové škálování analytického serveru a náklady ve formě udržování celého řešení dedikovaného analytického serveru včetně například zajištění migrace dat, bezpečnosti migrovaných dat a udržování případných licencí potřebných pro provoz platformy analytického serveru.

Jak již bylo uvedeno v kapitole čtvrté, oproti předcházejícím dvěma rozšiřujícím konceptům je tomuto rozšiřujícímu konceptu věnován v této práci menší rozsah. Hlavními cíli této kapitoly je vyzkoušet tento koncept v testovacím prostředí a popsat možnosti jeho využití na základě výstupů z testovacího prostředí a vlastních zkušeností autora.

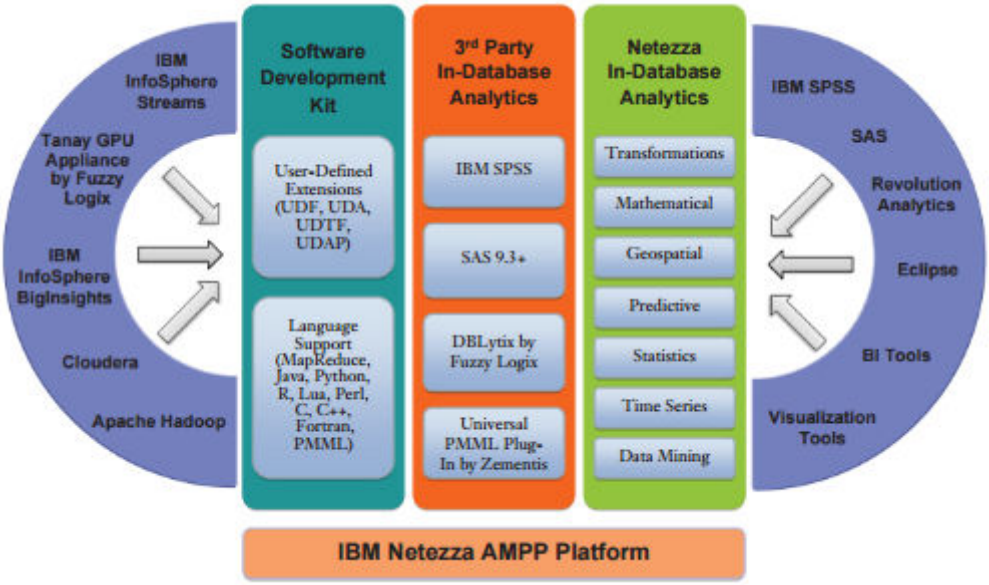
8.1 Analytické funkce a možnosti použití

Použití analytických funkcí v datovém skladu velmi úzce souvisí s použitou technologií, na které je daný datový sklad postaven. Pokud je to možné, měly by být již při výběru technologie datového skladu zohledněny požadavky na používání analytických funkcí. V praxi, kdy datový sklad je již postaven na starší technologii, která analytické funkce nepodporuje nebo podporuje velmi omezeně, často nezbývá jiné řešení, než přesunout náročné či specifické analytické výpočty na dedikovaný server.

Testovací prostředí, které je použité v této práci, využívá jako hlavní databázový systém technologii IBM Netezza. Konkrétně se jedná o Data Warehouse Appliance, která se celým jménem nazývá IBM® PureData™ for Analytics, powered by Netezza technology. Tento systém obsahuje rozšíření IBM® Netezza® Analytics, které je možné použít jako pokročilou analytickou platformu (IBM, 2017c).

Tato platforma nabízí mimo jiné vysoký výkon s využitím Netezza AMPP architektury (Asymmetric Massively Parallel Processing architecture). Také umožňuje využití

předpřipravených analytických funkcí, integraci s nástroji třetích stran a vytvoření vlastních analytických funkcí pomocí podporovaných jazyků (IBM 2017c). Tyto možnosti analytické platformy jsou přehledně znázorněny na následujícím obrázku číslo 19.



Obr. 19 – Znázornění možností analytické platformy IBM Netezza Analytics (IBM, 2017c)

Z těchto všech možností, které platforma nabízí, bylo pro vyzkoušení v testovacím prostředí vybráno použití programovacího jazyka R a C++. Jazyk R byl vybrán proto, že se jedná o Open Source řešení, které umožňuje pokročilou datovou analýzu a je jej možné rozšířit o mnoho komunitně vyvíjených balíčků, které zvětšují možnosti jeho použití. Obecně lze také říci, že se stává velmi oblíbeným a používaným (například dle (CASS, 2016) a (TIOBE, 2017)).

Jazyk C++ byl vybrán z toho důvodu, že se jedná o velmi rozšířený a obecně uznávaný programovací jazyk. S jeho pomocí by mělo být možné na platformě IBM Netezza Analytics rozšířit možnosti jazyka SQL, kdy funkce napsané v C++ budou volány přímo z jazyka SQL. I z důvodu vyzkoušení tohoto použití byl právě tento jazyk vybrán k vyzkoušení v testovacím prostředí.

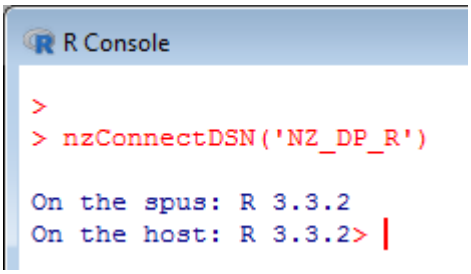
Jazyku R a jazyku C++ jsou věnovány následující dvě subkapitoly, kdy každá z nich se věnuje právě jednomu jazyku a jeho použitím v testovacím prostředí. Obě dvě tyto subkapitoly vychází z oficiální dokumentace (IBM, 2016a) a dostupných informací z IBM

portálu (IBM, 2017d). Vlastní výstupy této kapitoly vycházejí z výsledků a ze zjištění z testovacího prostředí a osobních zkušeností autora při používání těchto technologií.

8.1.1 Nasazení a vyzkoušení R v testovacím prostředí

Pro nasazení jazyka R do testovacího prostředí byl zvolen následující postup. Prvně byly staženy zdrojové kódy jazyka R, které byly zkompilovány v prostředí systému Netezza. Tyto zkompilované kódy byly následně včleněny do analytické platformy IBM Netezza Analytics. V samostatném prostředí na systému Windows byla nainstalována klasická distribuce R s nadstavbou RGui ve verzi 3.3.2. Do tohoto prostředí byly standardním způsobem přidány tři Netezza balíčky určené pro R, které zprostředkovávají komunikaci R Console s Netezza databází.

Po tomto nasazení je již možné se z R Console připojit do databázového systému Netezza. Toto připojení je znázorněno na následujícím obrázku číslo 20, kde jsou i zobrazené nainstalované verze jazyka R přímo na Netezze a jejích samostatných paralelizovaných výpočetních jednotkách (SPU). Jak je vidět na obrázku, na Netezze i na jejich SPU je naistalována stejná verze jazyka R 3.3.2.



Obr. 20 – Úspěšné připojení z R Console do databázového systému Netezza (zdroj: autor)

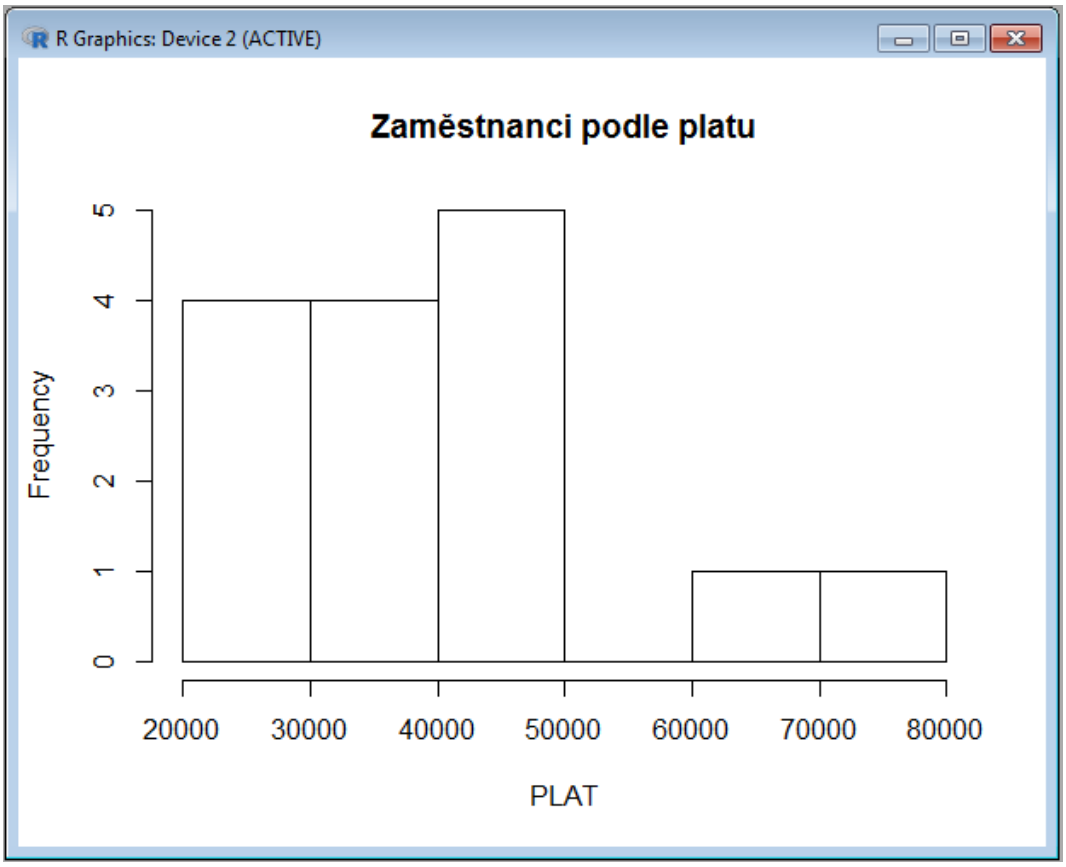
Nyní je již možné pracovat v nativním prostředí R Console s daty uloženými v databázi Netezza obdobně, jako v případě, kdy by byly uložené na klientské stanici nebo samostatném analytickém serveru. Rozdíl je v tom, že místo aby data byla přímo uložená na klientské stanici nebo na analytickém serveru, daná datová struktura, s kterou uživatel pracuje, obsahuje pouze ukazatel na danou strukturu v databázi. Samotný výpočet se poté realizuje přímo v databázi a je na uvážení uživatele, zda výsledky výpočtu chce stáhnout

do klientské stanice nebo opět uložit do databáze a například je použít pro další kroky v analytickém výpočtu.

Pro ověření funkčnosti tohoto propojení byly v databázi Netezza spočítány data pro vykreslení jednoduchého histogramu a vytvořen analytický model rozhodovacího stromu. Výsledky výpočtu byly staženy do prostředí RGui a vykresleny v tomto prostředí stejně tak, jako kdyby výpočet proběhl přímo zde. Podkladová data pro tyto výpočty vychází z předchozích dvou rozšiřujících konceptů, konkrétně se jedná o data z CDC, která již jsou zpracovaná ve vrstvě L1 pomocí Historizace. Samotný výpočet proběhl na aktuálních datech z posledního zpracování zpřístupněných pomocí jednoduchého databázového pohledu z vrstvy L1.

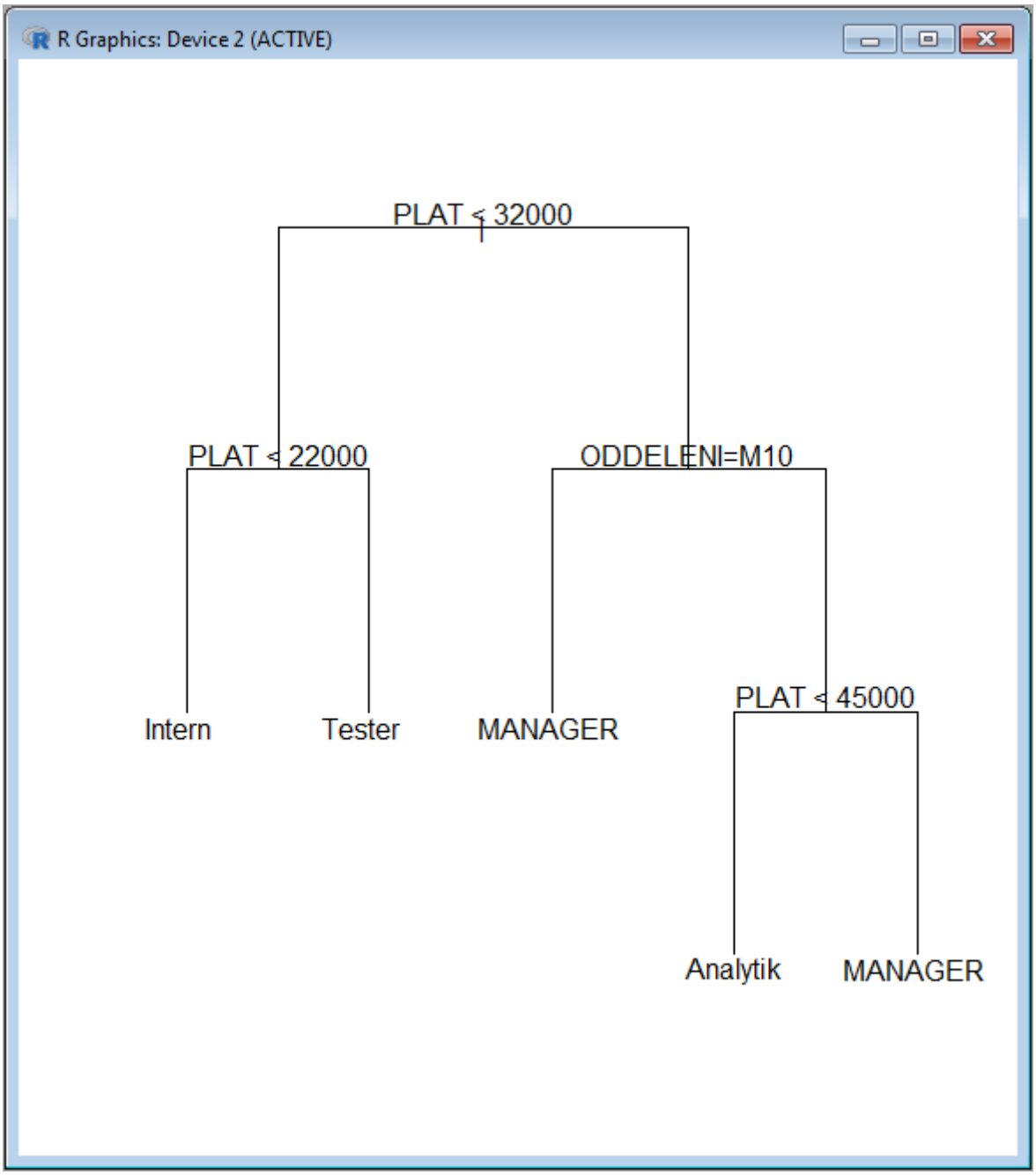
Toto použití ilustruje případné použití v reálném prostředí, kdy data ze zdrojových systémů jsou automaticky přenášena do cílového systému, kde jsou automaticky zpracovávána a nabídnuta uživatelům k analytickému využití. Vzhledem k tomu, že se jedná pouze o testovací data a tato práce si neklade za cíl je analyzovat, byla tato data ve vrstvě L1 záměrně upravena pro účely této kapitoly tak, aby výsledky a principy použití analytických funkcí bylo možné vhodně zachytit v této práci.

Vykreslený jednoduchý histogram je zobrazen na následujícím obrázku číslo 21. Tento histogram zobrazuje rozdělení zaměstnanců do skupin podle platů, tedy kolik zaměstnanců spadá do dané platové kategorie.



Obr. 21 – Histogram vypočítaný v databázi Netezza a zobrazený v prostředí R (zdroj: autor)

Na dalším obrázku číslo 22 je zobrazen model rozhodovacího stromu, vypočítaný v databázi Netezza na základě dat uložených přímo v tomto databázovém systému. Pouze výsledky nutné pro vykreslení rozhodovacího stromu byly opět staženy do prostředí RGui. Tento rozhodovací strom je modelem na základě podkladových dat, který slouží k určení pozice zaměstnance na základě jeho platu, oddělení a pohlaví.



Obr. 22 – Model rozhodovacího stromu vypočítaný v databázi Netezza a zobrazený v prostředí R (zdroj: autor)

Při interpretaci tohoto výsledku, který je zobrazen v modelu rozhodovacího stromu a založen pouze na testovacích datech, by bylo vhodnou otázkou, zda plat určuje pozici nebo je tomu naopak, tedy že pozice určuje výši platu. Tyto otázky souvisí s účelem modelu a jsou již předmětem práce datových analytiků a jsou mimo rámec této práce. Cílem tedy není interpretace testovacích dat. Znázorněný rozhodovací strom na obrázku číslo 22 slouží

pouze pro ilustraci datového stromu, který je vypočítán přímo v databázi s využitím masivně paralelního výpočtu a bez přesunu podkladových dat do jiné platformy je následně zobrazen v prostředí RGui stejně tak, jako by byl vypočítán přímo v tomto prostředí.

Jak v případě spočítaného histogramu, tak i v případě vytvořeného rozhodovacího stromu bylo možné veškeré úkony související s výpočtem provést z prostředí RGui. Z pohledu datových analytiků, kteří jsou zvyklí pracovat v prostředí RGui, tedy není nutné osvojovat si nové prostředí pro práci s databází či složitě provádět migraci dat do jiných analytických platforem.

8.1.2 Nasazení a vyzkoušení vlastních funkcí v testovacím prostředí

Tato subkapitola se věnuje možnostem vytvoření vlastních databázových funkcí, které je možné využívat pro analytické potřeby. Jak již bylo uvedeno, pro vytvoření těchto funkcí v databázi Netezza byl zvolen jazyk C++.

V Netezze je možné vytvořit několik typů funkcí, které lze poté využít například pro specifické analýzy. Tyto funkce opět využívají masivní paralelní zpracování dat, které platforma nabízí. Všechny tyto typy funkcí nesou v názvu anglický pojem User-defined, který již sám naznačuje, že se jedná o funkce definované uživatelem. Jednotlivé typy těchto uživatelských funkcí jsou (IBM, 2015b):

- User-defined function (UDF)
- User-defined table function (UDTF)
- User-defined aggregates (UDA)
- User-defined shared libraries

User-defined function, zkráceně UDF, je typ skalární funkce, tedy takové, která vrací jednu návratovou hodnotu. Tento typ funkce lze použít v rámci dotazu SQL kdekoli, kde je možné použít vestavěné skalární funkce.

User-defined table function, zkráceně UDTF, je typ funkce, která dokáže vrátit výsledek v podobě relační tabulky. Tato tabulka má předem pojmenované sloupce a určené typy. Tuto funkci je možné použít v rámci dotazu SQL téměř kdekoli, kde je možné použít klasickou relační tabulku.

User-defined aggregates, zkráceně UDA, je typ agregační funkce. Tento typ funkce může být použit s mnoha parametry, jedná se ale o skalární typ funkce, tedy takový typ funkce, který vrací pouze jednu hodnotu. Tento typ agregační funkce lze použít v dotazu SQL kdekoli, kde je možné použít vestavěný typ agregační funkce.

User-defined shared libraries jsou funkce, které není možné volat přímo z dotazů SQL. Tento typ funkcí slouží pro zjednodušení ostatních typů funkcí, kdy je možné vyčlenit společné části kódů do těchto User-defined shared libraries (sdílených knihoven). Z ostatních typů funkcí je poté možné volat takto vyčleněné funkce.

Pro všechny tyto uvedené typy funkcí se také používá souhrnné označení UDX, které zastřešuje všechny uživatelské funkce, tedy UDF, UDTF, UDA a User-defined shared libraries.

Pro vyzkoušení v testovacím prostředí byly zvoleny první dva uvedené typy funkcí, tedy UDF a UDTF. Pro jejich vývoj lze použít jakýkoli textový editor nebo vývojové prostředí. Pro vyvinutí těchto funkcí v rámci této práce bylo použito vývojové prostředí Eclipse. Vyvinuté funkce byly zkompilovány v prostředí systému Netezza a následně byly registrovány v testovací databázi. Od tohoto okamžiku je již možné vytvořené vlastní databázové funkce používat v rámci dotazů SQL.

První vyvinutou funkcí použitou pro vyzkoušení vlastních funkcí byla funkce UDF s názvem zpracuj_retezec. Vstupním parametrem této funkce je textový řetězec. Tento vstupní řetězec je ve funkci zpracován tak, že je rozdělen na menší řetězce podle znaku středníku. Tyto jednotlivé jeho části jsou přetypovány na čísla a tyto čísla jsou následně sečtena. Výsledná hodnota je vrácena jako výstup funkce. Následující obrázek číslo 23 zobrazuje použití této funkce v rámci dotazu SQL, včetně vstupních dat, které jsou pro vyzkoušení funkce použita. Jak je z obrázku patrné atribut str1, který obsahuje textový řetězec, vstupuje jako parametr do funkce zpracuj_retezec, která provede popsané zpracování a v tomto případě vrátí výsledek v novém sloupci a to pro každý záznam, který je vybrán v rámci dotazu SQL.

```
select ID, str1, zpracuj_retezec(str1) from data_fce1 order by ID;
```

Output | Result 1

Drag a column header here to group by that column.

	ID	STR1	ZPRACUJ_RETEZEC
1	1	2;9	11
2	2	5;5;1;2;6	19
3	3	1;2;3;4;5;6	21
4	4	999;999;19	2017

Obr. 23 – Zpracování textového řetězce pomocí nově vytvořené funkce zpracuj_retezec (zdroj: autor)

Z tohoto obrázku číslo 23 je také patrná přidaná hodnota takto vytvořených nových funkcí. V momentě kdy takováto funkce je vytvořena, lze jí opakovaně použít pro specifické potřeby při práci s daty. Její používání je relativně jednoduché a také relativně transparentní vzhledem k tomu, že tato funkce je volána přímo z dotazů SQL. Takto vytvořené funkce pro specifické účely také mohou nabídnout funkcionalitu, které pomocí dotazů SQL a vestavěných databázových funkcí nemusí být možné vůbec dosáhnout.

Druhou vyvinutou funkcí, která byla použita pro vyzkoušení vlastních funkcí v testovacím prostředí, byla funkce vytvor_tab. Tato nová funkce je typu UDTF a vstupuje do ní pět parametrů. Výstupem této funkce je podoba relační tabulky se čtyřmi sloupci, kdy pro každou sadu vstupných parametrů vrací celkem čtyři řádky.

Vstupní data použitá pro vyzkoušení této funkce jsou zobrazena na obrázku 24, který je uveden níže. Jedná se o relační tabulku data_fce2, která má tři záznamy a pět sloupců. První z nich je sloupec ID, který slouží jako primární klíč tabulky. Další čtyři sloupce (ATR1 až ATR4) obsahují čtyři hodnoty, které slouží pro výpočet.

```
select * from data_fce2 order by id;
```

Output | Result 1

Drag a column header here to group by that column.

	ID	ATR1	ATR2	ATR3	ATR4
1	1	6	2	10	2
2	2	10	10	20	20
3	3	10	5	100	2

Obr. 24 – Relační tabulka obsahují vstupní data pro nově vytvořenou funkci typu UDTF (zdroj:autor)

Všechny sloupce této tabulky data_fce2 vstupují do nově vytvořené funkce vytvor_tab. Tato funkce poté vrací čtyři nové sloupce. První je sloupec REF, který je cizí klíč a tvoří referenci do zdrojové tabulky. Druhý a třetí sloupec jsou vypočítané sloupce, kdy na základě druhého a třetího parametru pro druhý sloupec a třetího a čtvrtého parametru pro třetí sloupec jsou postupně provedeny základní aritmetické operace. Jaký typ aritmetické operace byl proveden, obsahuje poslední čtvrtý sloupec TYP, který textovým řetězcem tuto provedenou operaci pojmenovává.

Pro každou sadu vstupních parametrů jsou provedeny čtyři základní aritmetické operace, odečtení, sečtení, vydělení a vynásobení. Funkce tedy vytvoří čtyři nové řádky pro každý vstupní řádek. Volání funkce, včetně předání vstupních dat jako parametrů a výsledek funkce ve formě relační tabulky je zobrazen na následujícím obrázku číslo 25.

```
select vysledek.*
from data_fce2, table(vytvor_tab(id, atr1,atr2,atr3, atr4)) vysledek
order by id, typ;
```

Output | Result 1

Drag a column header here to group by that column.

	REF	I	J	TYP
1	1	4	8	odecteno
2	1	8	12	secteno
3	1	3	5	vydeleno
4	1	12	20	vynasobeno
5	2	0	0	odecteno
6	2	20	40	secteno
7	2	1	1	vydeleno
8	2	100	400	vynasobeno
9	3	5	98	odecteno
10	3	15	102	secteno
11	3	2	50	vydeleno
12	3	50	200	vynasobeno

Obr. 25 – Výsledek ve formě relační tabulky vlastní funkce UDTF včetně jejího volání (zdroj: autor)

Jak je na tomto obrázku 25 vidět, funkce vytvor_tab funguje dle očekávání. Ze vstupních dat ze sloupce ATR1 a ATR2 jsou vybrány hodnoty na kterých je provedena aritmetická operace vypsaná ve sloupci TYP a výsledek této operace je vrácen ve sloupci I. To samé platí pro data ze sloupce ART3 a ATR4, kdy je na těchto hodnotách provedena stejná aritmetická operace a výsledek je vrácen do sloupce J.

Tato konkrétní funkce nemá specifickou přidanou hodnotu a slouží zejména pro demonstraci možností tohoto typu funkce. Jak je na tomto příkladu vidět, nově vytvořená tabulka si může pomocí reference zachovat vazbu na zdrojovou tabulku. S touto dynamicky vytvořenou tabulkou je možné dále pracovat přímo pomocí jazyka SQL, například jí použít v dalších výpočtech. Tuto tabulku je také možné materializovat například jednoduše pomocí jazyka SQL a jeho konstrukcí jako jsou dotazy typu „CREATE TABLE AS“ nebo „INSERT INTO SELECT“. Tento materializovaný výsledek může být poté použit v dalších výpočtech nebo přímo jako vstup do nástrojů pro tvorbu reportů.

8.2 Zhodnocení konceptu

Jak bylo potvrzeno v rámci předchozích dvou subkapitol, při zvolení vhodné technologie datového skladu je možné použít analytické funkce přímo v technologii datového skladu. Jak bylo také ukázáno na příkladech s využitím jazyka R, analytický výpočet probíhal přímo v technologii datového skladu i přes to, že uživatel mohl pracovat v obvyklém prostředí určeném pro tento jazyk. To poskytuje určitou výhodu v tom, že datový analytik se může soustředit na svou práci v prostředí, v kterém se dobře orientuje a má s ním již zkušenosti, bez toho aby se musel starat o to, kde se data nacházejí, zda jsou správně přesunuta, či zda má všechna data, která potřebuje pro výpočet, aktuální na dedikované analytické platformě.

V rámci vyzkoušení vlastních analytických funkcí bylo potvrzeno, že je možné vyvinout funkce pro velmi specifický účel, které je možné transparentně volat z dotazovacího jazyka SQL. Tyto funkce byly vyvinuty v jazyce C++ a mohou nabídnout funkcionality pro složité analytické úlohy, které by jinak vůbec nebylo možné dosáhnout nebo případně velmi složitě s možnými negativními výkonovými dopady.

Použití tohoto konceptu v rámci datového skladu může mít určité výhody. Například se jedná o sdílený výkon, kdy není potřeba výkonově škálovat dvě technologické platformy a řešit integraci mezi nimi. Je možné takto ušetřené finanční prostředky investovat právě do jedné technologie, což může přinést flexibilitu v rámci přesunů výpočetního výkonu podle aktuálních priorit, kterými může být například zpracování dat, reporting či právě pokročilá analytika. Také například pokud nastane mimořádná situace a je potřeba akutně zpracovat určitá data, lze výkon alokovat právě na tuto činnost. V případě dedikovaného analytického serveru by datový sklad při takového situaci nemusel mít dostatečný výkon na poskytnutí aktuálních dat dedikovanému analytickému serveru, což by mohlo způsobit, že jeho výkonová kapacita by nemohla být využitá.

Dalším přínosem je, že všechna data v datovém skladu jsou ihned k dispozici pro analytické použití. Jak bylo ukázáno v testovacím prostředí, není potřeba provádět migraci či jejich úpravu. Data byla v případě použití R dostupná přes R objekt, který obsahoval ukazatel na datovou strukturu v databázi, a v případě vlastních funkcí byla data dostupná přes parametr, který do těchto funkcí vstupoval.

Za určitou výhodu lze v tomto konkrétním případě považovat i to, že rozšíření IBM Netezza Analytics, které slouží jako pokročilá analytická platforma, je dodáváno s databází IBM Netezza a v rámci celého produktu IBM PureData for Analytics, powered by Netezza technology je v době psaní této práce dostupné bez potřeby dodatečných licencí.

Při použití tohoto konceptu v reálném prostředí je potřeba zvážit dodatečné požadavky na výpočetní výkon, které vzniknou při použití technologie datového skladu i pro analytické potřeby. Při volbě konkrétní technologie a jejího výpočetního výkonu je tedy potřeba do odhadu zahrnout kromě obvyklých výkonových požadavků, které jsou kladeny na datový sklad i požadavky související se specifickými analytickými úlohami, které zde budou zpracovávané. Jak již bylo zmíněno v předcházejícím odstavci, při vhodném využívání výpočetního výkonu jednotné technologie lze dosáhnout určité flexibility, která může být užitečná při nestandardních situacích. Vzhledem ke zkušenostem autora této práce lze doporučit při použití tohoto konceptu navrhnout a provozovat vhodné řízení výpočetních zdrojů (tzv. Database Resource Management) právě s ohledem na analytické potřeby a jejich dopady do vytížení celého systému.

Při zkoušení tohoto rozšiřujícího konceptu v testovacím prostředí se v některých případech objevovaly v databázi dočasné objekty, které obsahovaly data potřebná pro výpočet, například pro tvorbu konkrétních modelů. Tyto objekty zůstávaly v databázi i po provedení výpočtu. Při praktickém použití tohoto konceptu je tedy potřeba prověřit toto chování na konkrétní zvolené technologii a případně zařadit čištění těchto dočasných objektů mezi pravidelně prováděnou údržbu.

Při použití vlastních definovaných funkcí, které rozšiřují možnosti zvolené platformy, je vhodné k nově vyvinutým funkcím vytvářet řádnou dokumentaci. Jak bylo ukázáno na příkladech vytvořených funkcí v C++, tyto funkce lze volat přímo z dotazovacího jazyka SQL. Z názvů těchto funkcí nemusí být zřejmý jejich účel a algoritmus, který provádějí. Detailně vytvořená dokumentace, včetně popisu jednotlivých vstupních parametrů, poté zvýší transparentnost používání takto vytvořených funkcí v rámci dotazů jazyka SQL.

Za možnou důležitou výhodu tohoto konceptu lze označit relativně jednoduchou přenositelnost ze stávajících řešení, které jsou již v organizaci používané pro analytické úlohy. S touto situací se autor práce přímo v praxi již setkal a jedná se například o situace, kdy společnost již má analytické kódy či celá analytická řešení vyvinuté v technologii nebo

jazyce (například v této práci použité R nebo C++), které obsahují unikátní logiku či know-how. Pokud analytická platforma datového skladu podporuje tyto technologie nebo jazyky je jejich převod relativně jednoduchý. V podstatě se jedná jen o napojení na dané rozhraní poskytované danou platformou, v případě testovacího prostředí konkrétně platformou IBM Netezza Analytics. S relativně nízkou pracností lze tak řešení přesunout do technologie datového skladu a využívat tak již zmíněné výhody tohoto rozšiřujícího konceptu.

Při zvažování tohoto přesunu nebo při výběru nové dodatečné technologie v rámci datového skladu pro analytické použití je možné v praxi téměř vždy doporučit provést podobné vyzkoušení na konkrétních příkladech tak, jako bylo provedeno v rámci této kapitoly. Může se jednat například o malý projekt, jehož cílem by bylo potvrzení konceptu a jeho přínosů (Proof of concept) vzhledem k potřebám konkrétního analytického použití v dané společnosti. Zejména se jedná o případy, kdy je již například dané analytické oddělení specializováno na používání určitého analytického nástroje. Přeučení a adaptace zcela nové technologie by poté mohla být velmi časově i finančně náročná. Pokud ale nová technologie umožňuje vysokou míru možné integrace, tak jako bylo ukázáno v rámci této kapitoly, přechod a kombinace daných technologií by mohla být vysoce přínosná. Zmíněný projekt pro potvrzení konceptu v reálném prostředí by tak mohl výrazně snížit rizika a určitou míru nejistoty, která existuje při akvizici nové technologie.

9. Závěr

Tato kapitola obsahuje zhodnocení splnění stanovených cílů a krátce shrnuje výsledky a zjištění, kterých bylo dosaženo v této práci. Je zde také krátce popsán obsah jednotlivých kapitol, které se přímo neváží k jednotlivým rozšiřujícím konceptům, a to zejména vzhledem k definovaným cílům této práce.

9.1 Zhodnocení splnění stanovených cílů

Cílem této diplomové práce bylo analyzovat rozšiřující koncepty použitelné v datovém skladu, vyzkoušet je v testovacím prostředí a s využitím poznatků z praxe popsat možné přínosy, nevýhody, rizika a možnosti uplatnění těchto konceptů v prostředí datového skladu. Pro splnění tohoto hlavního cíle byly definovány následující tři dílčí cíle.

1) Definovat a zdůvodnit výběr konkrétních rozšiřujících konceptů datových skladů zvolených pro vyzkoušení v testovacím prostředí.

2) Vytvořit testovací prostředí a vyzkoušet použití rozšiřujících konceptů na konkrétních technologiích společnosti IBM.

3) Analyzovat dopady použití rozšiřujících konceptů na datový sklad zejména vzhledem k možným přínosům při jejich použití, případně analyzovat alternativní řešení.

Po úvodní kapitole se v této práci nachází kapitola druhá, která se zabývá komentovanou rešerší informačních zdrojů. Cílem této kapitoly je provést rešerši a nalézt vhodné zdroje, které by mohly být přínosné pro tuto práci a případně nalézt další odborné články, závěrečné práce, a příspěvky z odborných konferencí, které se zabývají podobným tématem jako tato diplomová práce.

Následuje kapitola třetí, která popisuje základní teorii datových skladů a základy architektury datových skladů a poté vymezuje rozšiřující koncepty tak, jak s nimi dále pracuje tato diplomová práce.

Tyto dvě kapitoly tak vytváří nezbytný základ a východiska pro splnění dílčích cílů i hlavního cíle této práce. Splněním dílčích cílů a hlavního cíle se poté zabývají kapitoly, které po těchto kapitolách následují.

Prvnímu dílčímu cíli se věnuje kapitola čtvrtá Výběr rozšiřujících konceptů. Vybrané rozšiřující koncepty jsou v této kapitole popsány a je zdůvodněno, proč byly právě tyto rozšiřující koncepty vybrány k analýze a k vyzkoušení v testovacím prostředí. Celkem byly vybrány tři rozšiřující koncepty Zachycení změn ve zdrojových systémech, Historizace a Analytické funkce a datový sklad. Tím byl první dílčí cíl této práce splněn.

Druhým dílčím cílem se zabývalo více kapitol. Kapitola pátá se věnovala technologiím a produktům, které byly použity pro vytvoření testovacího prostředí. Všechny tyto technologie a produkty jsou zde uvedeny a je zde popsána technologie Netezza, která tvořila základ testovacího prostředí. Samotné testovací prostředí bylo vytvářeno postupně v rámci kapitol, které se věnovaly jednotlivým rozšiřujícím konceptům, konkrétně se jedná o kapitoly šest, sedm a osm. Tento zvolený postup umožnil, aby technologie nebo produkty přímo související s daným rozšiřujícím konceptem byly uvedeny přímo u tohoto konceptu včetně jejich začlenění do testovacího prostředí. Subkapitoly, které se věnují zejména tomuto začlenění do testovacího prostředí, nesou název Nasazení do testovacího prostředí, konkrétně se jedná o kapitoly 6.4, 7.4, 8.1.1 a 8.1.2. Vyzkoušením rozšiřujících konceptů na konkrétních technologiích se věnují kapitoly, které nesou název Vyzkoušení konceptu v testovacím prostředí, konkrétně se jedná o kapitoly 6.6, 7.5, 8.1.1 a 8.1.2 a jejich související subkapitoly. V rámci těchto uvedených kapitol bylo vytvořeno funkční testovací prostředí a všechny zvolené rozšiřující koncepty v něm byly vyzkoušeny. Tím byl i tento dílčí cíl splněn.

Třetím dílčím cílem se také zabývalo více kapitol. Každý rozšiřující koncept a jeho dopady byly analyzovány v rámci kapitoly, která se věnovala konkrétnímu rozšiřujícímu konceptu. Základní zjištění a výstupy jsou uvedeny v kapitolách věnujících se vyzkoušení rozšiřujících konceptů v testovacím prostředí. Hlavní část analýzy a popis dopadů při použití rozšiřujících konceptů je uveden v kapitolách, které nesou název Zhodnocení konceptu, konkrétně se jedná o kapitoly 6.7, 7.6 a 8.2. V těchto kapitolách jsou zjištění a výstupy z testovacího prostředí shrnuty a dány do širšího kontextu datového skladu, zejména vzhledem k možným přínosům, kterých může být dosaženo při jejich použití v datovém skladu. Tím byl tento třetí dílčí cíl splněn.

Hlavním cílem se zabývaly kapitoly věnující se konkrétním rozšiřujícím konceptům, především se pak jedná o podkapitoly, které celé koncepty zhodnocovaly. Všechny tři zvolené rozšiřující koncepty byly v testovacím prostředí vyzkoušeny a analyzovány. U všech tří analyzovaných rozšiřujících konceptů byly také popsány možné přínosy, nevýhody, rizika a možnosti uplatnění těchto konceptů v prostředí datového skladu, kdy tento popis byl obohacen o autorovy zkušenosti z praxe.

Jak vyplývá z předcházejících čtyř odstavců, hlavní cíl i všechny dílčí cíle této práce byly splněny.

9.2 Shrnutí výsledků a zjištění

Tato podkapitola shrnuje nejpodstatnější výsledky, které byly zjištěny v rámci této práce. Nejedná se o kompletní výčet výsledků a zjištění. Kompletní výčet výsledků a zjištění je možné nalézt v rámci daných kapitol, které se věnují daným rozšiřujícím konceptům, a to včetně detailního popisu a zdůvodnění.

9.2.1 Zachycení změn ve zdrojových systémech (CDC)

V rámci prvního rozšiřujícího konceptu Zachycení změn ve zdrojových systémech, byly vyzkoušeny tři hlavní typy mapování, které určovaly výslednou formu zachytávaných dat. Dva z těchto typů mapování tvořily v cílovém systému obraz zdrojového systému, který umožňoval práci s daty v cílovém systému obdobně, jako kdyby se přistupovalo k datům ve zdrojovém systému.

Třetí typ mapování LiveAudit umožňoval zaznamenávat pouze změny, které proběhly na zdrojovém systému. Pomocí vhodného nastavení technických polí a s využitím Java třídy se v testovacím prostředí podařilo dosáhnout stavu, kdy z těchto identifikovaných změn lze jednoznačně určit stav dat zdrojového systému v libovolném minulém časovém bodě a to i v případě, kdy na zdrojovém systému proběhne více změn naráz v rámci stejné databázové transakce v rychlém sledu za sebou během jedné sekundy.

Z této ověřené vlastnosti bylo vyvozeno, že s pomocí tohoto mapování lze efektivně zpracovat data do historické kolekce dat a také zpracovávat data zdrojových systémů k

přesně určeným časovým bodům například pro potřeby reportingu. Tyto tvrzení byly ověřeny v testovacím prostředí v rámci druhého rozšiřujícího konceptu Historizace. Shrnutí všech výsledků a zjištění souvisejících s prvním rozšiřujícím konceptem lze nalézt v kapitole 6.7 Zhodnocení konceptu.

9.2.2 Historizace

Druhý rozšiřující koncept se věnoval historizaci dat. V testovacím prostředí bylo vyvinuto a vyzkoušeno zpracování dat do historické kolekce dat, kdy vstupními daty byla data dostupná z mapování LiveAudit z předchozího rozšiřujícího konceptu CDC. Tato data tedy obsahovala pouze změny provedené na zdrojovém systému.

Na základě zjištění z testovacího prostředí bylo navrženo generické řešení zpracování dat ze zdrojových systémů, které je znázorněno na obrázku číslo 16 na straně 59. V testovacím prostředí bylo také zprovozněno a vyzkoušeno zpracování dat k určitému času, včetně tzv. zpětného zpracování dat k určitému minulému času s využitím auditních dat. Princip zpětného zpracování je zobrazen na obrázku číslo 17 na straně 63, kde je i detailně vysvětlen a popsán na konkrétních testovacích datech.

Obdobně jako u předchozího rozšiřujícího konceptu, shrnutí všech výsledků a zjištění souvisejících s tímto druhým rozšiřujícím konceptem lze nalézt v kapitole 7.6 Zhodnocení konceptu.

9.2.3 Analytické funkce a datový sklad

Třetí rozšiřující koncept se věnoval použití analytických funkcí v technologii datového skladu. V testovacím prostředí bylo vyzkoušeno, že je možné pracovat v nativním analytickém prostředí RGui a přenést samotný výpočet k datům, umístěným v datovém skladu, bez nutnosti jejich migrace. Pro demonstraci těchto možností byl v databázi Netezza spočítán jednoduchý histogram a vytvořen model rozhodovacího stromu.

V rámci tohoto rozšiřujícího konceptu byly také vyvinuty dvě nové analytické funkce v jazyce C++, které bylo možné použít z dotazovacího jazyka SQL. Bylo tedy potvrzeno, že je možné vyvinout vlastní analytické funkce pro specifické analytické účely a získat tak novou unikátní funkcionalitu.

Obdobně jako v předchozích případech souhrn všech výsledků a zjištění souvisejících s tímto třetím rozšiřujícím konceptem je uveden v kapitole 8.2 Zhodnocení konceptu.

Terminologický slovník

Termín	Zkratka	Význam (Zdroj)
Asymmetric Massively Parallel Processing	AMPP	Asymetrické masivně paralelní zpracování, které využívá technologie Netezza. Toto zpracování využívá přístup, kdy jednotlivé výkonové části jsou na sobě nezávislé (shared nothing approach) (Francisco, 2014).
Change Data Capture	CDC	Zachycení pouze relevantních změn na zdrojových systémech (Kimball, 2013).
Data Staging Area	DSA	Viz Dočasné uložiště.
Datastore (CDC)		Část produktu IBM InfoSphere Change Data Capture (CDC). Jsou označovány jako instance, které obsahují datové soubory a samotný replikační engine. Z pohledu používání se jedná o kontejnery, které umožňují pracovat se zdrojovými a cílovými systémy (IBM, 2016d).
Datový sklad	DWH, DW	Systém, který extrahuje, čistí, přizpůsobuje a dodává zdrojová data do dimenzionálního datového uložiště a poté podporuje a implementuje dotazování a analýzy pro účely rozhodování (Kimbal, 2004). (viz kapitola 3.1)
Dočasné uložiště	DSA	Uložiště, které slouží k dočasnému uložení extrahovaných dat z produkčních systémů. Data v tomto uložišti jsou detailní, neagregovaná, často nekonzistentní, bez časové dimenze (Pour, et al., 2012).
Field Programmable Gate Array (Netezza)	FPGA	V rámci technologie Netezza se jedná o specializovaný výpočetní čip, který je uzpůsobený pro výrazné zrychlení základních databázových operací (Francisco, 2014).
Historizace		Proces zpracování dat do datové kolekce, která obsahuje časovou dimenzi, resp. je možné v této kolekci určit, kdy která data byla či jsou platná (autor) (viz kapitola 7).
Rozšiřující koncept datového skladu		Koncept datového skladu, který rozšiřuje možnosti datového skladu tím, že buď přidává novou funkcionalitu, nebo řeší jeho určitý generický problém (autor) (viz vymezení v kapitole 3.3).
Snippet Blades	S-Blades	Základní výkonová jednotka masivně paralelního systému Netezza (Francisco, 2014).

Termín	Zkratka	Význam (Zdroj)
Snippet Processing Units	SPU	Synonymum k S-Blades, který je uveden v tomto slovníku výše.
Structured Query Language	SQL	Jazykové rozhraní pro relační systémy (Inmon a Linstedt, 2015).
Subscriptions (CDC)		Logické kontejnery, které obsahují detaily o datech, která jsou přenášena mezi zdrojovým a cílovým datastore. Také obsahují nastavení, jak mají být zdrojová data uložena do cílového systému (IBM, 2016d).
Temporální data		Data o stavu objektů v minulosti, přítomnosti a budoucnosti a/nebo tvrzeních o tom, co data znamenala v minulosti, přítomnosti a budoucnosti (Johnston a Weis, 2010).
Transformační nástroje	ETL	Extraction, Transformation, Load nebo také označováno jako datová pumpa. Hlavním úkolem ETL je přenos dat z jednoho či více systémů (často zdrojových systémů), transformace dat do nové vhodné podoby a nahrání těchto dat do cílového systému nebo systémů (Pour, et al., 2012).
User-defined aggregates	UDA	Typ agregační funkce. Tuto funkci lze vytvořit v C++ a rozšířit tak funkcionalitu databáze Netezza (IBM, 2015b).
User-defined function	UDF	Typ skalární funkce. Tuto funkci lze vytvořit v C++ a rozšířit tak funkcionalitu databáze Netezza (IBM, 2015b).
User-defined table function	UDTF	Typ funkce, která dokáže vrátit výsledek v podobě relační tabulky. Tuto funkci lze vytvořit v C++ a rozšířit tak funkcionalitu databáze Netezza (IBM, 2015b).
User-defined shared libraries		Tento typ funkcí slouží pro zjednodušení ostatních typů funkcí UDX, kdy je možné vyčlenit společné části kódů do těchto User-defined shared libraries (sdílených knihoven) (IBM, 2015b).
UDX	UDX	Souhrnné označení pro UDF, UDTF, UDA a User-defined shared libraries (IBM, 2015b).

Použité informační zdroje

CASS, Stephen, 2016. The 2016 Top Programming Languages: C is No. 1, but big data is still the big winner. IEEE Spectrum [online]. IEEE Spectrum [cit. 2017-02-12]. Dostupné z: http://spectrum.ieee.org/computing/software/the-2016-top-programming-languages

DVORSKÝ, Bohuslav, 2016. Využití Big Data v bankovním prostředí. Praha. Diplomová práce. Vysoká škola ekonomická v Praze.

FRANCISCO, Phil, 2014. IBM PureData System for Analytics Architecture: A Platform for High Performance Data Warehousing and Analytics [online]. IBM [cit. 2017-01-19]. REDP-4725-01. Dostupné z: http://www.redbooks.ibm.com/redpapers/pdfs/redp4725.pdf

Gartner, 2016. Data Lake. Gartner: IT Glossary [online]. Gartner, 2016 [cit. 2016-12-20]. Dostupné z: http://www.gartner.com/it-glossary/data-lake/

HAVLAS, Petr, 2015. Vytvoření datového skladu. Praha. Diplomová práce. Bankovní institut vysoká škola Praha, a.s. Vedoucí práce Doc. Ing. Bohumil Miniberger, CSc.

HEJMALÍČEK, Aleš, 2015. Hadoop as an Extension of the Enterprise Data Warehouse. Brno. Diplomová práce. Masarykova univerzita, Fakulta informatiky. Vedoucí práce Doc. RNDr. Vlastislav Dohnal, Ph.D.

HNÍK, Pavel, 2011. Provoz a udržitelný rozvoj datového skladu. Praha. Diplomová práce. Vysoká škola ekonomická v Praze.

IBM, 2010. IBM to Acquire Netezza. IBM News room [online]. United States: IBM [cit. 2016-08-14]. Dostupné z: http://www-03.ibm.com/press/us/en/pressrelease/32514.wss#release

IBM, 2015a. InfoSphere Data Replication V11.3.3 documentation. IBM Knowledge Center [online]. IBM Corporation [cit. 2016-10-25]. Dostupné z: https://www.ibm.com/support/knowledgecenter/SSTRGZ_11.3.3/com.ibm.idr.frontend.doc/pv_welcome.html

IBM, 2015b. IBM Netezza User-Defined Functions Developer's Guide: IBM Netezza. Rev. 3. USA: IBM Corporation 2007, 2015.

IBM, 2016a. PureData System for Analytics V7.2.1 documentation. IBM Knowledge Center [online]. IBM Corporation [cit. 2016-10-25]. Dostupné z: https://www.ibm.com/support/knowledgecenter/SSULQD_7.2.1/com.ibm.nz.welcome.doc/kc_welcome_V721.html

IBM, 2016b. InfoSphere Change Data Capture: Replicate information in your heterogeneous data stores. IBM Software [online]. IBM [cit. 2016-10-05]. Dostupné z: http://www-03.ibm.com/software/products/en/ibminfochandatacapt

IBM, 2016c. Supported source and targets. IBM Knowledge Center [online]. IBM [cit. 2016-10-15]. Dostupné z: https://www.ibm.com/support/knowledgecenter/SSTRGZ_11.3.3/com.ibm.cdcdoc.sysreq.doc/concepts/supportedsourceandtargets.html

IBM, 2016d. About CDC Replication. IBM Knowledge Center [online]. IBM [cit. 2016-10-12]. Dostupné z: https://www.ibm.com/support/knowledgecenter/SSTRGZ_11.3.3/com.ibm.cdcdoc.sysreq.doc/concepts/aboutcdc.html

IBM, 2016e. Preparing your environment for an installation of the CDC Replication Engine for Netezza Technology. IBM Knowledge Center [online]. IBM [cit. 2016-10-16]. Dostupné z: https://www.ibm.com/support/knowledgecenter/en/SSTRGZ_11.3.3/com.ibm.cdcdoc.cdcfornetezza.doc/concepts/allocatinganothermachineforcdc.html

IBM. 2016f. Using journal control fields for auditing replication activities. IBM Knowledge Center [online]. IBM [cit. 2016-11-07]. Dostupné z: https://www.ibm.com/support/knowledgecenter/en/SSTRGZ_11.3.3/com.ibm.cdcdoc.mcadminguide.doc/concepts/aboutjournalcontrolfields.html

IBM, 2016g. Replicating Data Definition Language (DDL) changes. IBM Knowledge Center [online]. IBM [cit. 2016-11-12]. Dostupné z: https://www.ibm.com/support/knowledgecenter/en/SSTRGZ_11.3.3/com.ibm.cdcdoc.mcadminguide.doc/concepts/replicatingddlchangesoverview.html

IBM, 2017a. DB2 11.1 for Linux, Unix and Windows documentation. IBM Knowledge Center [online]. IBM Corporation, [cit. 2016-10-25]. Dostupné z: https://www.ibm.com/support/knowledgecenter/SSEPGG_11.1.0/com.ibm.db2.luw.kc.doc/welcome.html

IBM, 2017b. IBM Netezza Data Warehouse Appliances: The Simple Data Warehouse Appliance for Serious Analytics [online]. IBM, [cit. 2017-01-05]. Dostupné z: https://www-01.ibm.com/software/data/netezza/

IBM, 2017c. IBM PureData System: IBM Netezza Analytics. IBM - Netezza Analytics [online]. United States: IBM [cit. 2017-02-12]. Dostupné z: https://www-01.ibm.com/software/data/puredata/analytics/nztechnology/analytics.html

IBM, 2017d. IBM PureData-Netezza Developer Network (NDN) [online]. IBM [cit. 2017-01-12]. Dostupné z: https://www.ibm.com/developerworks/community/groups/service/html/communityview?communityUuid=266888e9-4b4b-44cd-bd51-e32d05da9143

INMON, William H. a Daniel LINSTEDT, 2015. DATA ARCHITECTURE: A PRIMER FOR THE DATA SCIENTIST: Big Data, Data Warehouse and Data Vault. USA: Elsevier. ISBN 978-012-8020-449.

INMON, William H, 2005. Building the data warehouse. 4th ed. Indianapolis, Ind.: Wiley. ISBN 07-645-9944-5.

JOHNSTON, Tom a Randall WEIS, 2010. Managing time in relational databases: how to design, update and query temporal data. 1. Boston: Morgan Kaufmann/Elsevier. ISBN 01-237-5041-5.

KIMBALL, Ralph a Margy ROSS, 2013. The data warehouse toolkit: the definitive guide to dimensional modeling. Third edition. Indianapolis, Indiana: John Wiley & Sons. ISBN 978-1-118-53080-1.

KIMBALL, Ralph a Joe CASERTA, 2004. The Data Warehouse ETL Toolkit: Practical Techniques for Extracting, Cleaning, Conforming, and Delivering Data. 1. Indianapolis, IN: Wiley Publishing. ISBN 07-645-7923-1.

KUŽELKA, Kryštof, 2015. Datový sklad v prostředí Amazon Web Services. Praha. Diplomová práce. Vysoká škola ekonomická v Praze.

LOPEZ, Karen a Joseph D'ANTONI, 2014. The Modern Data Warehouse-How Big Data Impacts Analytics Architecture. Business Intelligence Journal. 19(3), 8-15.

NOVOTNÝ, Ota, Jan POUR a David SLÁNSKÝ, 2005. Business intelligence: Jak využít bohatství ve vašich datech. Praha: Grada. Management v informační společnosti. ISBN 80-247-1094-3.

POUR, Jan, Miloš MARYŠKA a Ota NOVOTNÝ, 2012. Business intelligence v podnikové praxi. Praha: Professional Publishing. ISBN 978-807-4310-652.

PLAČEK, Ondřej, 2016. Datový sklad pro analýzu finanční a učetní oblasti v rámci informačního systému QI. Brno. Diplomová práce. Mendelova univerzita v Brně, Provozně ekonomická fakulta. Vedoucí práce Ing. Jan Přichystal, Ph.D.

ROKYTA, Jan, 2013. Datový sklad k technologiím budov. Brno. Diplomová práce. Masarykova univerzita, Fakulta informatiky.

ŠŤAVA, Michal, 2015. Robustní a škálovatelný datový sklad pro systém Perun. Brno. Diplomová práce. Masarykova univerzita, Fakulta informatiky.

ŠILER, Zdeněk, 2012. Vývoj datového skladu na platformě Teradata a Informatica v sektoru pojišťovnictví. Praha. Diplomová práce. Vysoká škola ekonomická v Praze.

TANK, Darshan M., Amit GANATRA, Y.P. KOSTA a C.K. BHENSDADIA, 2010. Speeding ETL Processing in Data Warehouses Using High-Performance Joins for Changed Data Capture (CDC). In: 2010 International Conference on Advances in Recent Technologies in Communication and Computing. IEEE, s. 365-368. DOI: 10.1109/ARTCom.2010.63. ISBN 978-1-4244-8093-7. Dostupné také z: http://ieeexplore.ieee.org/document/5656810/

TIOBE, 2017. The R Programming Language. TIOBE - The Software Quality Company [online]. TIOBE [cit. 2017-02-12]. Dostupné z: http://www.tiobe.com/tiobe-index/r/

ZHOU, Xin, Fusheng WANG a Carlo ZANIOLO, 2006. Efficient Temporal Coalescing Query Support in Relational Database Systems. In: Database and Expert Systems Applications: 17th International Conference, DEXA 2006, Kraków, Poland, September 4-8, 2006. Proceedings. Kraków, Poland,: Springer Berlin Heidelberg, s. 676. DOI: 10.1007/11827405_66. ISBN 978-3-540-37871-6. ISSN 0302-9743. Dostupné také z: http://link.springer.com/10.1007/11827405_66

Seznam obrázků

Seznam tabulek