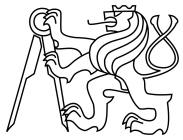
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ



ZADÁNÍ DIPLOMOVÉ PRÁCE

Název:	Návrh datových vrstev pro datový sklad ČVUT
Student:	Bc. Jakub Krejčí
Vedoucí:	Ing. Magda Friedjungová
Studijní program:	Informatika
Studijní obor:	Webové a softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2017/18

Pokyny pro vypracování

1) Seznamte se s problematikou datového skladu a proveďte rešerši používaných architektur.
2) Navrhněte architekturu vrstvy s integrovanými daty (tzv. integrated data layer) a přístupové vrstvy (tzv. access layer) datového skladu ČVUT.
3) Proveďte analýzu jmenných konvencí v oblasti databází a navrhněte metodiku pro jmennou konvenci databázových objektů v datovém skladu ČVUT.
4) Navrhněte datový model databáze s integrovanými daty a integrujte do ní zdrojové systémy KOS, Portál pro závěrečné práce (BPM FIT) a Anketa ČVUT. Při návrhu využijte metodiku pro jmennou konvenci.
5) Tento datový model využijte pro vytvoření vrstvy s integrovanými daty.
6) Navrhněte a implementujte sémantickou vrstvu.
7) Pro účely analýzy pomocí technologie OLAP vytvořte přístupovou vrstvu (tzv. access layer) datového skladu ČVUT s vhodnými datovými tržišti (tzv. data marts). Demonstrujte využití této technologie na několika příkladech a shrňte výhody a nevýhody takto prezentovaných dat.

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

prof. Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 22. prosince 2016

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

Diplomová práce

Návrh datových vrstev pro datový sklad ČVUT

Bc. Jakub Krejčí

Vedoucí práce: Ing. Magda Friedjungová

8. května 2017

Poděkování

Chtěl bych poděkovat své vedoucí diplomové práce **Ing. Magdě Friedjungové** odborné vedení, pomoc a rady při zpracování této práce.

Dále bych chtěl poděkovat **Bc. Robertu Kotlářovi**, **Ing. Stanislavu Kuznetsovovi** za spolupráci na projektu datového skladu ČVUT a vedoucímu tohoto projektu za vedení a přístup **Ing. Michalu Valentovi, PhD.**.

Nakonec bych rád poděkoval všem blízkým a příbuzným za pochopení a podporu.

Prohlášení

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

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 8. května 2017 .....

Odkaz na tuto práci

Krejčí, Jakub. *Návrh datových vrstev pro datový sklad ČVUT* . Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2017.

Abstrakt

Tato diplomová práce se zabývá návrhem a implementací datových vrstev datového skladu ČVUT, který vznikl rámci rozvojových projektů (DÚ č. 20 a č. 46). V práci je popsána teorie z oblasti business intelligence, datových skladů, architektur datových skladů a databázových jmenných konvencí. V implementační části je popsán návrh databázové jmenné konvence pro datový sklad a architektura datového skladu ČVUT. V rámci práce byla provedena implementace jednotlivých datových vrstev a tvorba datového modelu centrální databáze. Tato implementace byla otestována vizualizací datového tržiště pomocí nástroje Pentaho BI Server.

Klíčová slova Datový sklad, datový model, jmenná konvence, centrální databáze, datové tržiště, integrovaná datová vrstva, přístupová vrstva.

Abstract

This master's thesis describes the design and implementation of a data layer solution for the CTU data warehouse, which is being implemented in development projects (DÚ n. 20 a n. 46). The thesis describes the theory of business

intelligence, data warehouses, data warehouse architectures and database naming conventions. The implementation part consists of the design of a database naming convention for the data warehouse architecture of the CTU data warehouse. It describes the implementation of individual data layers and the design of a central database. This implementation was tested using datamart visualisation in Pentaho BI Server.

Keywords Data warehouse, data model, central database, datamart, Integrated Data Layer, Access Layer.

Obsah

Seznam obrázků

Seznam tabulek

Úvod

V této diplomové práci se věnuji tématu datových skladů a implementaci datového skladu v rámci rozvojových projektů (DÚ č. 20 a č. 46). Základní technická funkcionalita datového skladu ČVUT byla rozdělena na dvě části. Datovou integraci se zabývá Robert Kotlář ve své diplomové práci [1]. Já se věnuji návrhu datových vrstev datového skladu ČVUT.

V teoretické části jsou popsány pojmy a krátká historie spojená s business intelligence, aby bylo možné lépe pochopit důvody existence datových skladů. Následně jsou vysvětleny a definovány základní pojmy spojené s datovými sklady. V další části jsou rozebrány architektury datových skladů ze dvou různých pohledů (Inmonnův a Kimballův) a je zde vysvětlena architektura, která je nejvíce používaná v dnešní době.

Dále se také věnuji databázové jmenné konvenci. Jmenná konvence je zadefinována, probrána základní problematika a její výhody vysvětleny na ukázkových dotazech psaných v jazyce SQL.

Praktická část této práce nejprve popisuje jmennou konvenci, kterou jsem vyvinul pro datový sklad ČVUT. Následně popisuji vývoj centrální databáze datového skladu. Jsou zde vysvětleny jednotlivé modely a kroky, které jsem při návrhu datového modelu databáze využil.

Dále se věnuji architektuře datového skladu ČVUT. Především se zde zaměřuji na datové vrstvy architektury. K těmto vrstvám je v práci vysvětlen způsob implementace.

Na konec se zabývám vytvářením obchodních entit na sémantické databázi a tvorbě datových tržišť. Nad datovými tržišti je provedena ukázková vizualizace dat pomocí nástroje Pentaho BI Server.

Cíl práce

Hlavním cílem práce bylo navržení datových vrstev tak, aby navržené řešení bylo snadno rozšiřitelné a dostatečně bezpečné a zároveň rychlé pro efektivní

vytváření analytických dotazů.

Sekundárním cílem práce bylo vytvoření jmenné konvence databáze, která zlepší orientaci v databázi datového skladu a bude jednoznačně definovat význam jednotlivých tabulek a atributů.

Část I

Teoretická část

Kapitola 1

Datové sklady a business intelligence

V první kapitole této diplomové práce se budu věnovat vysvětlení základních pojmů v oblasti datových skladů a business intelligence. Nejprve vysvětlím základní principy a historii business intelligence. Následně se budu zabývat spojitostí tohoto pojmu s datovým skladem.

V případě datových skladů stručně vysvětlím celý koncept a význam tohoto systému. Poté Vás seznámím se stručnou historií spojenou s datovými sklady. Následně představím dvě základní architektury datových skladů (dle Billa Innmona a dle Ralpha Kimballa).

Po tomto úvodu do problematiky datových skladů popíši architekturu, která se používá v současnosti. Na této architektuře vás již detailněji seznámím s jednotlivými součástmi moderního datového skladu a jejich funkcemi v rámci celého systému. Tato moderní architektura také poslouží k demonstraci možného způsobu získávání dat ze zdrojových systémů a k demonstraci historizace záznamů v datovém skladu.

Na konci této kapitoly se také zmíním o pojmech, které s datovými sklady přímo souvisí a v implementační části této diplomové práce jsou využívány. Jedná se o zvláštní způsob databázového modelování takzvané dimenzionální modelování a o technologii pro zpracování velkého objemu dat OLAP (viz sekce 1.8).

1.1 Business intelligence

Business intelligence (zkráceně BI) je v dnešní době velice populární a velmi diskutovaný pojem, jak v akademické sféře (především v ohledu na narůstající možnosti při ukládání a zpracovávání dat i ze systémů obsahujících nestrukturované informace), tak v komerční sféře, která do BI technologií investuje stále více prostředků.

Tento pojem je i přes jeho současnou popularitu často špatně pochopen a definován. V této diplomové práci používám definici L. Gály [2], která zní:

> „Business intelligence je sada procesů, know-how, aplikací a technologií, jejichž cílem je účinně a účelně podporovat řídící aktivity ve firmě. Podporují analytické plánovací a rozhodovací činnosti organizací na všech úrovních a ve všech oblastech podnikového řízení, tj. prodeje, nákupu, marketingu, finančního řízení, controllingu, majetku, řízení lidských zdrojů, výroby a dalších."

Tato definice nám popisuje business intelligence jako sadu různých částí (od procesů až po technologie), které nám pomáhají sledovat dění ve firmě, vyhodnocovat jednotlivé problémy a také do určité míry předvídat budoucnost firmy. I v jiných odborných publikacích autoři [3] [4] často pracují s obdobnou definici, občas se také můžeme setkat s alternativními definicemi [5], které jsou svázány s informačními technologiemi (především s datovými sklady), což je špatná interpretace tohoto pojmu. Samotné principy business intelligence jsou dosažitelné i bez informačních technologií, ale dosáhnutí podobného výsledku jako za použití informačních technologií je velmi náročné, neefektivní a drahé (manuálně získávat, zpracovávat, ukládat a vyhodnocovat informace ve firmě).

1.1.1 Historie

Principy spojené s business intelligence se začaly používat s rozvojem podnikání a byly používány k zvýšení zisku jednotlivých podnikatelů. Hospodářské dějiny definují zlom v podnikání přechodem z cechovního řemesla na výrobu v manufakturách. Tato výroba se začala rozvíjet v 15. a 16. století v severní Itálii a Flandrech. Jelikož tato forma výroby zefektivňovala a zjednodušovala výrobní procesy (již tu nebyl vztah mistr – tovaryš, ale vztah bližší dnešní době, tedy podnikatel – námezdní dělník), docházelo k nárůstů produktivity práce a zvýšení objemu zhotovených výrobků. Nárůstem počtu zaměstnanců, zvyšováním objemu obchodu a zvyšováním tržeb bylo nutné začít vyhodnocovat a archivovat informace o počtu a stavu zaměstnanců, objemu tržeb v jednotlivých obdobích a další informace užitečné pro konkrétního podnikatele za účelem získání konkurenční výhody. Tento způsob řízení manufaktur můžeme pokládat za první masovější rozšíření principů dnešní business intelligence. [6]

Pojem business intelligence byl poprvé zmíněn v roce 1865 v knize Cyclopaedia of Commercial and Business Anecdotes Richardem Devensem [7]. V této knize se autor zmiňuje o bankéři Henrym Furnesovi jehož výhoda oproti ostatním spočívá ve shromažďování informací, které mu pomáhají v jeho podnikání. Devens zde toto shromažďování informací definuje jako business intelligence: „Throughout Holland, Flanders, France and Germany, he maintained a complete and perfect train of business intelligence".

V odborné literatuře byl pojem business intelligence nejprve definován pracovníkem IBM Hansem Peterem Luhnem ve vědeckém článku publikovaném

roku 1958 [8]. Autor v tomto článku popisuje automatický systém vyrobený za účelem zpracování, správy a šíření velkého množství informací v organizacích.

Vývoj business intelligence ve formě, ve které je nám známa v současnosti začal v 60. letech 20. století a dosáhl vrcholu v 80. letech. V této době začaly být vhodnou alternativou pro ukládání informací ve společnostech pevné disky počítačů. Tato forma elektronického ukládání dat vedla k rozvoji DSS[1] a DMS[2].

V roce 1989 Howard Dresner použil termín business intelligence, jako pojmenování tohoto nového konceptu vzniklého z DSS v průběhu minulých desetiletích (protože tyto systémy a procesy s nimi spojené již vysoce převyšovaly vlastnosti spojené s běžně používanými DSS systémy). V 90.letech se také začaly objevovat první business intelligence systémy a nástroje, které již vyvíjeli společnosti IBM a Oracle [9]. Systémy vzniklé v této době bývají označené jako generace BI 1.0.

Od roku 2000 se již se setkáváme s business intelligence označovanou jako BI 2.0, která využívá cloudu, narůstajícího výpočetního výkonu a možností rychle přenášet informace za použití internetu.

V dnešní době využíváme mobilní přístup k jednotlivým prvkům BI a zpracováváme exponenciálně narůstající množství dat (tento nárůst je i důsledek většího důrazu na zpracování tzv. big data). Také v současnosti není business intelligence doménou pouze velkých nadnárodních společností, ale našla svou cestu i do řízení velmi malých podniků.

1.1.2 Spojitost datových skladů s BI

Často jsou data potřebná pro business intelligence získávána z datových skladů nebo z datových tržišť, která jsou součástí architektury datového skladu. Ale všechny datové sklady nejsou využívány pro business intelligence a každá implementace business intelligence nemusí využívat datový sklad [10]. Toto chování plyne ze sedmi základních vlastností, které by měla implementace business intelligence splňovat.

> „In general, BI provides a framework for:
>
> 1. Collecting and storing operational data.
> 2. Aggregating the operational data into decision support data.
> 3. Analyzing decision support data to generate information.
> 4. Presenting such information to the end user to support businnes decisions.

[1] Systém na podporu rozhodování (anglicky Decision Support System) pomáhá uživatelům při řízení a rozhodování ve společnosti.

[2] Systém pro správu a oběh dokumentů (anglicky Document Management System slouží ke správě elektronických dokumentů nebo zdigitalizovaných papárových dokumentů ve společnosti.

5. Making business decisions, which in turn generate more data that are collected, stored, and so on (restarting the process).
6. Monitoring results to evaluate outcomes of the business decisions, which again provides more data to be colected, stored and so on.
7. Predicting future behaviors and outcomes with a high degree off accuracy.“ [11]

Datový sklad je technologie, která implementuje první dvě základní vlastnosti business intelligence, protože datový sklad slouží k ukládání dat z jednotlivých provozních informačních systémů organizací (první vlastnost) a také umožňuje vytvářet různé agregované pohledy nad daty uloženými do datového skladu (druhá vlastnost). Tyto dvě první vlastnosti mohou být implementovány i jiným způsobem, ale pro současné potřeby organizací je datový sklad nejefektivnější řešení.

1.2 Datové sklady

V organizacích se vyskytují tři základní typy dat. Jsou to data operační (někdy označovaná i jako transakční), data operativní a data analytická [12].

- *Operační data* (anglicky transactional data) jsou vytvářena provozními informačními systémy organizace a slouží k ukládáních záznamů o aktivitách v obchodních procesech. Tato data zpravidla nejsou sdílena do ostatních provozních systémů organizace.
- *Operativní data* (anglicky master data) přiřazují k transakčním datům vazby k obchodním entitám (zákazníci, dodavatelé, zaměstnanci). Tato data jsou většinou sdílena mezi jednotlivými operačními systémy organizace a bývají zpravidla persistentní.
- *Analytická data* (anglicky analytical data) podporují především rozhodování v organizacích. Shlukují operační data do vyšších celků (různé stupně agregace nad operačními daty).

Systém, který integruje operační data jednotlivých informačních systémů organizace do jednotného celku společně s operativními daty, umožňuje zachovávat hodnoty, které jednotlivé atributy historicky nabývaly a nad těmito daty je schopen vytvářet analytická data, se nazývá datový sklad.

V datových skladech existují dva základní pohledy na celkový koncept – pohled dle Billa Inmonna a pohled dle Ralpha Kimballa, kteří jako první začali první implementovat a vyvíjet datové sklady. Oba jsou v současnosti označování jako otci datových skladů.

Odlišnosti plynoucí z těchto dvou různých pohledů jsou patrné již z definic, viz. níže. Tyto rozdíly budou detailněji probrány nad jimi definovanými architekturami datových skladů v sekci 1.4 této diplomové práce.

Definice datového skladu dle Billa Inmonna je následující:

> „A data warehouse is a subject-oriented, integrated, nonvolatile, and time-variant collection of data in support of management's decisions.“[13]

Bill Inmon definuje datový sklad charakteristikami, které splňuje a kterými se odlišuje od ostatních analytických systémů.

Datový sklad je subjektově orientován (**subject-oriented**) na konkrétní oblast, kterou se organizace zabývá. Například pro výrobní společnost jsou klíčové entity produkt, objednávka, prodejce, účet a surovina.

Další a nejdůležitější charakteristikou je integrovanost dat uložených v datovém skladu (**integrated**). Data jsou ukládána ze zdrojových provozních systémů (mnohdy zde dochází k duplicitám nebo odlišnému pohledu na význam ukládaných dat) do jednotného pohledu na organizaci, který je stanoven fyzickým databázovým modelem hlavní databáze datového skladu.

Data uložená v datovém skladu jsou stálá (**nonvolatile**) – data se v datovém skladu neodstraňují ani neaktualizují. Data se do datového skladu nahrávají jako stav provozního systému k určitému datu. Pokud se při následném nahrání dat do datového skladu (stav provozního systému z následujícího dne) záznam aktualizuje nebo odstraní, tak je vytvořen nový záznam nebo záznamu je ukončena platnost – rozhodně záznam získaný při minulém nahrání dat do datového skladu není aktualizován nebo není odstraňován.

Poslední zásadní charakteristikou datového skladu je časový rozptyl dat, které uchovává (**time-variant**). Tato vlastnost souvisí s předchozí vlastností, která zaručuje, že historický záznam nemůže být přemazán novějším záznamem. Datový sklad tedy uchovává v sobě i historické stavy provozních systémů. Zpravidla takto uchováváme historické stavy po dobu 5-10 let (může se lišit z legislativních důvodů nebo potřebami jednotlivých organizací).

Naproti tomu definice datového skladu dle Ralpha Kimballa je následující:

> Data warehouse is „the queryable source of data in the enterprise“ a technicky ho můžeme definovat jako „the union of all the constituent data marts“. [14]

Tato Kimballova definice popisuje datový sklad jako zdroj dat pro analýzu a reporting ve společnostech. Říká, že datový sklad je sjednocení všech datových tržišť (data mart – segment datového skladu, který obsahuje informace pro analýzu určité části společnosti například určitý úsek výroby, zároveň musí

být v Kimballově architektuře reprezentován pomocí dimenzionálního modelu, viz sekce 1.5).

1.2.1 Historie

V 80. letech 20. století začaly vznikat systémy podporující rozhodování (DSS) diametrálně odlišné od provozních nebo transakčních systémů organizací. Pro část, která obsahovala potřebná data pro chod těchto analytických aplikací, bylo potřeba vyvinout oddělená datová úložiště, a proto byla navrhnuta architektura s nezávislými datovými tržišti [15]. V těchto letech začíná Bill Inmon definovat a diskutovat pojem datový sklad v odborných kruzích [16].

V průběhu 90. let začínají především velké telekomunikační, finanční společnosti vytvářet první datové sklady. V tomto období také vznikají dva základní pohledy na datové sklady, tedy Inmonův a Kimballův a je představen databázový počítač DBC/1012 pro zpracování analytických dat v systémech podporujících rozhodování od společnosti Teradata [16].

V následujících letech se zlepšuje architektura datových skladů, ve které bývají umístěny i NoSQL technologie. Vznikají specializované databázové systémy (např. Teradata) poskytující vysoký výkon pro analytické zpracování velkého množství dat (často jsou zpracovávána i tzv. Big Data).

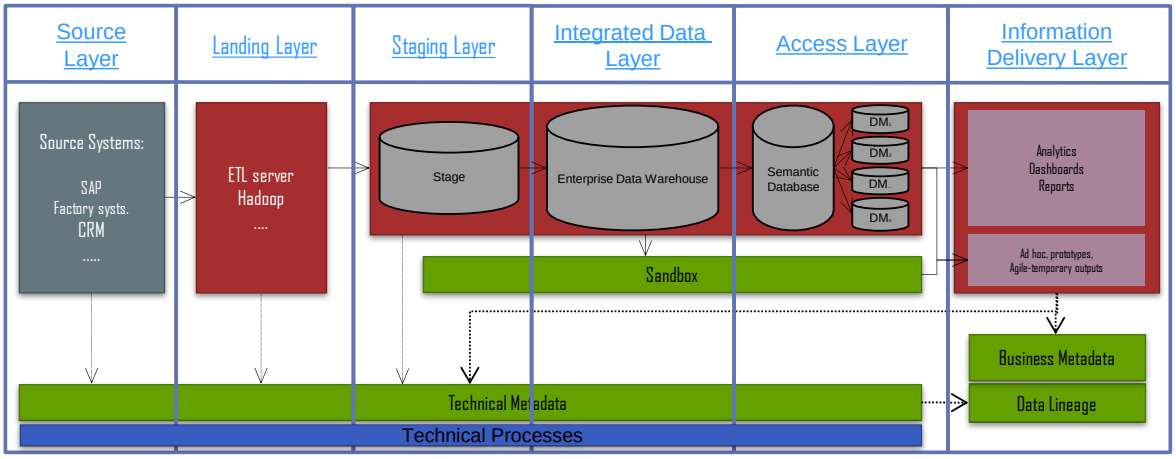
1.3 Současná architektura datových skladů

V současnosti se používají různé typy architektur datových skladů. Většina používaných architektur vychází z architektury navržené Ralphem Kimballem nebo Billem Inmonem (tyto architektury jsou rozebrány v následující části – sekce 1.4). Nejprve popisuji tuto současnou architekturu, protože obsahuje všechny části, které jsou použité i v základních architekturách. Jednotlivé části a pojmy mohu tedy vysvětlit na jednom příkladu architektury, což je přehlednější.

Architekturu datového skladu je nutné volit dle konkrétních požadavků a specifikací na implementaci. Na obrázku 1.1 se nachází architektura, která se používá v současných skladech větších institucí nebo v institucích s většími požadavky na implementované řešení (klíčový prvek podnikové informační infrastruktury, velké množství zdrojových systémů, očekávaný velký rozpočet na rozvoj business intelligence a podobně).

Tato architektura vychází z Inmonovy architektury datového skladu (také nazývané jako hub and spoke). Na rozdíl od Inmonovy architektury je rozšířena o takzvanou sémantickou vrstvu (semantic database), která definuje obchodní termíny a pravidla přímo nad databází, ze které jsou vytvářena jednotlivá datová tržiště (jednotné definice obchodních pojmů = „jednotná pravda“).

V následujících podsekcích stručně popíši význam a funkci jednotlivých vrstev architektury datového skladu zobrazených na obrázku 1.1.



Obrázek 1.1: Současná architektura datových skladů [17]

1.3.1 Source Layer

První vrstvou datového skladu jsou zdrojové provozní informační systémy (ERP – Enterprise Resource Planning, CRM – Customer Relationship Management atd.). Tyto systémy pracují kromě sdílení určitých obchodních entit (operativní data – např. zákazníci, dodavatelé) nezávisle. Z tohoto důvodu mají unikátní datový model, který je spjat pouze s určitými implementovanými obchodními procesy. Datový model zdrojových systémů nikdy nepokryje kompletně potřeby celé instituce.

Z těchto zdrojových systémů se získávají data (extract) do datového skladu v nezměněné podobě. Je nutné získávání dat vyřešit jednoduše a optimálně, aby tento proces příliš nezatěžoval zdrojový systém a uživatelům neznemožňoval jeho používání. Z tohoto důvodu se v případě, že data do datového skladu nemusí být ukládána v reálném čase, odehrává proces získávání dat mimo pracovní dobu uživatelů (většinou noční hodiny).

1.3.2 Data Integration Layer

Vrstva datové integrace může obsahovat dvě části – Landing Layer a Staging Layer. V případě, že ze zdrojových systémů získáváme datové extrakty formou, kterou musíme ještě upravovat před nahráním do databáze s aktuálním obrazem zdrojového systému umístěné ve Staging Layer (textové exporty, změnové vektory apod.) musí architektura datového skladu obsahovat i Landing Layer. Jinak v této vrstvě postačuje pouze Staging Layer.

1.3.2.1 Landing Layer

Tato vrstva slouží pro dodávání dat do celé architektury datového skladu ze zdrojových systémů. Datový otisk zdrojového systému je dodán administrátory přímo na určené místo umístěné v této vrstvě. Data mohou být relačního i nerelačního charakteru a může být dodáván úplný otisk zdrojového systému nebo datový inkrement (nové, změněné, smazané záznamy) oproti minulému otisku zdrojového systému.

V této vrstvě dochází především ke kontrole, zda byly potřebné soubory správně dodány a mají správnou předem definovanou strukturu. Po těchto kontrolách a případném zrekonstruování dat (pokud se například nejedná o úplný datový otisk stavu zdrojovoého systému) jsou data nahrána do databáze umístěné ve Staging Layer.

1.3.2.2 Staging Layer

Staging Layer obsahuje databázi, do které je nahrán 1:1 otisk dat ze zdrojového systému. Ve Staging Layer může být struktura zdrojového systému rozšířena o technické atributy, které slouží pro kontrolu dat nebo pro následné nahrání dat do databáze umístěné ve vrstvě Integrated Data Layer (například časy nahrání datového extraktu do Staging Layer). Rozhodně se nesmí měnit struktura dat nebo hodnoty dat obsahujících obchodní logiku.

Data ze zdrojových systémů do databáze obsažené ve Staging Layer mohou být nahrána přímo pomocí transformací pro získání datových extraktů (obdoba ETL – pouze nedochází k žádné transformaci dat) nebo můžeme využívat nahrání datových extraktů nepřímo přes Landing Layer.

Hlavní účel této vrstvy je zpracování datových extraktů získaných ze zdrojových systémů a kontrola jejich správnosti – zejména z pohledu technické kvality (primární klíče, cizí klíče, databázové omezení). Tyto kontroly jsou velmi důležité před nahráním dat do databáze obsažené v IDL, protože v případě problémů s datovou kvalitou máme možnost reagovat na problémy již v ETL procesech (nedojde k nahrání dat z problémového zdrojového systému, data budou automaticky vyčištěna apod.). Touto reakcí minimalizujeme možnost nahrání dat, které nám mohou porušit integritu databáze ve vrstvě IDL.

1.3.3 Integrated Data Layer

> „Hlavním účelem vrstvy Integrated Data Layer (zkráceně IDL) je ukládat a udržovat data ve správné struktuře, která umožňuje vytváření analýz, vytváření jiných datových struktur s obchodním významem a umožňuje podporovat vytváření nových služeb pro obchodní uživatele.“ [17]

Integrated Data Layer obsahuje jednu databázi často označovanou jako „Enterprise Data Warehouse“ [17]. Tato databáze obsahuje integrovaná histo-

rická data z rozdílných zdrojových systémů používaných v instituci. Všechna tato rozdílná data jsou uložena v jednotné logické struktuře (datový model popisující celou instituci, který definuje datové struktury jedné databáze).

Datový model této centrální databáze by měl být nezávislý vůči zdrojovým systémům, ale měl by zároveň obsahovat všechny klíčové datové struktury zdrojových databází. Z těchto důvodů nám plynou následující požadavky na implementaci datového modelu:

- *Jmenná konvence* pomáhá jednotně definovat názvy databázových objektů. Tato metodika zvyšuje přehlednost databáze pro administrátory i analytiky. Více se této problematice věnuji v kapitole 2 této diplomové práce.

- *Subjektově orientovaný datový model* nám umožňuje zaměřit se při návrhu na konkrétní datové entity spojené se zaměřením instituce, pro kterou datový sklad implementujeme. Datový model centrální databáze datového skladu není modelován pro podporu obchodních procesů instituce, ale pouze za účelem spojení jednotlivých klíčových datových entit získaných ze zdrojových systémů. Takto vytvořený datový model by tedy neměl popisovat obchodní model instituce, ale vztahy mezi jednotlivými datovými entitami (v této centrální databázi se můžeme oprostit od obchodních omezení).

- *Normalizovaný datový model dle 3NF*[3] použijeme při tvorbě datového modelu centrální databáze. Oproti klasickým OLTP[4] databázím pro potřeby této centrální databáze postačí datový model, který se bude blížit splnění 3NF (musíme počítat s možnými anomáliemi z důvodu integrace více zdrojových systémů). Takto normalizovaná centrální databáze pomáhá k lepší implementaci obchodních pravidel definovaných v sémantické databázi umístěné v Access Layer. Takto normalizovaný datový model odstraňuje i datové redundance.

- *Škálovatelnost* je důležitý požadavek při tvorbě datového modelu centrální databáze datového skladu, protože při vývoji datového skladu musíme počítat s možnou integrací dalších zdrojových systémů.

Do datových struktur definovaných těmito požadavky nahráváme ETL procesy data s historickou hloubkou. Tato data bývají neshlukovaná (non-aggregated) – většinou se jedná o jednotlivé záznamy z tabulky umístěné ve

[3]Třetí normální forma je splněna pokud tabulka splňuje druhou normální formu (zkráceně 2NF, je splněna pokud všechny atributy tabulky jsou atomické a pokud každý atribut, který není primárním klíčem je na primárním klíči úplně závislý) a žádný atribut, který není primárním klíčem není tranzitivně závislý na žádném klíči.

[4]Online Transaction Processing je technologie pro ukládání dat v databázi, která ji využívána v provozních systémech institucí s mnoha uživateli.

zdrojovém systému. V případě této architektury datového skladu používáme metodiku pro ukládání historických dat dle historizace typu SCD 2 (viz refscd-Typu2). Tento přístup nám umožňuje vytvářet libovolné analytické dotazy a sledovat historické trendy dle zadefinovaných obchodních požadavků.

Nevýhodou tohoto přístupu je obrovský nárůst počtu záznamů uložených v centrální databázi v čase (musíme si uvědomit, že ukládáme veškeré historické změny jednotlivých záznamů).

1.3.4 Access Layer

V centrální databázi umístěné v IDL jsou data ukládána bez většího obchodního významu (viz popis předchozí vrstvy). Je tedy nutné tento obchodní význam někde definovat. Toto je hlavním účelem přístupové vrstvy (Access Layer), která vytváří datové struktury s obchodním významem podle vymezených obchodních pravidel. Takto vytvořené datové struktury se dodávají do Information Delivery Layer, odkud jsou dostupné koncovým uživatelům.

Přístupová vrstva se dělí na dvě hlavní části – sémantickou databázi (někde označovanou jako sémantickou vrstvu) a na jednotlivá datová tržiště.

1.3.4.1 Semantic Database

V této části se definují nad daty uloženými v centrální databázi jednotlivá obchodní omezení. Tímto postupem nám vznikají datové struktury popisující obchodní entity v instituci.

Je nutné si uvědomit, že v centrální databází dochází k integraci mnoha zdrojových systémů a proto je pro nás klíčové ji oprostit od obchodního významu – změny v chápání jednotlivých datových struktur dle obchodních pravidel nám mohou zničit integritu databáze. Proto je výhodné obchodní pravidla/omezení definovat nad integrovanými daty právě až v této části přístupové vrstvy.

Nejjednodušeji je možné obchodní pravidla/omezení implementovat pomocí pohledů nebo materializovaných pohledů nad centrální databází. Tímto krokem vzniknou nové datové struktury, které již obsahují obchodní logiku a zároveň oproti datovým strukturám obsažených v centrální databázi umístěné v IDL nemusí být normalizované.

1.3.4.2 Data marts

Česky datová tržiště obsahují informace spojené s určitým obchodním požadavkem. V přístupové vrstvě se nachází více datových tržišť splňující různé analytické obchodními požadavky. Schéma těchto datových tržišť je většinou vytvořeno za použití dimenzionálního modelování (star/snowflake schéma).

Data pro datová tržiště se získávají nad datovými strukturami s obchodní logikou definovanými v sémantické databázi. Můžeme vytvořit pro jednotlivá datová tržiště klasické databázové tabulky a data nahrát pomocí ETL procesů

nebo vytvořit materializované pohledy nad sémantickou databází (z důvodu udržení nízké doby odezvy databáze a složitější datové transformaci se zde již nevyplatí použít klasické databázové pohledy).

Datová tržiště vytvořená z dat ze sémantické vrstvy nám zaručují takzvanou „jednotnou pravdu“ – nemůže nastat situace, kdy dvě datové tržiště implementující podobné obchodní požadavky pracují s jinými daty („jinou verzí pravdy“). Protože veškerá data sloužící pro potřeby datových tržišť jsou získána ze sémantické databáze, kde dochází k samotnému vytváření datových struktur pro jednotlivé obchodní entity – nad datovými tržišti se již žádná obchodní logika nedefinuje.

1.3.5 Information Delivery Layer

Tato vrstva reprezentuje doručování informací k většině obchodních uživatelů (do ostatních vrstev přistupují většinou pouze odborní uživatelé – analytici, znalostní inženýři apod.). Může se jednat o různé typy datových výstupů nebo využívání dat umístěných v datovém skladu jinými aplikacemi:

- Vygenerované datové sestavy,
- napojení na analytické nástroje,
- napojení na dataminingové nástroje,
- dashboardy a interaktivní grafy,
- získávání dat pro další systémy (např. CRM).

Mezi nástroje, které můžeme použít pro vytváření obsahu nad daty uloženými v datovém skladu pro širší spektrum obchodních uživatelů, například patří:

- Tableau – vizualizace dat,
- Power BI – vizualizace dat,
- Pentaho BI Server – portál s BI společnosti,
- IBM Cognos – vizualizace dat, interaktivní dashboardy, možnost analyzování dat, celý portál s BI společnosti, tvorba ETL,
- Oracle Business Intelligence – vizualizace dat, interaktivní dashboardy, možnost analyzování dat, celý portál s BI společnosti.

V této části diplomové práce byly shrnuty základní vlastnosti a účel vrstev současné architektury datového skladu znázorněné na obrázku 1.1.

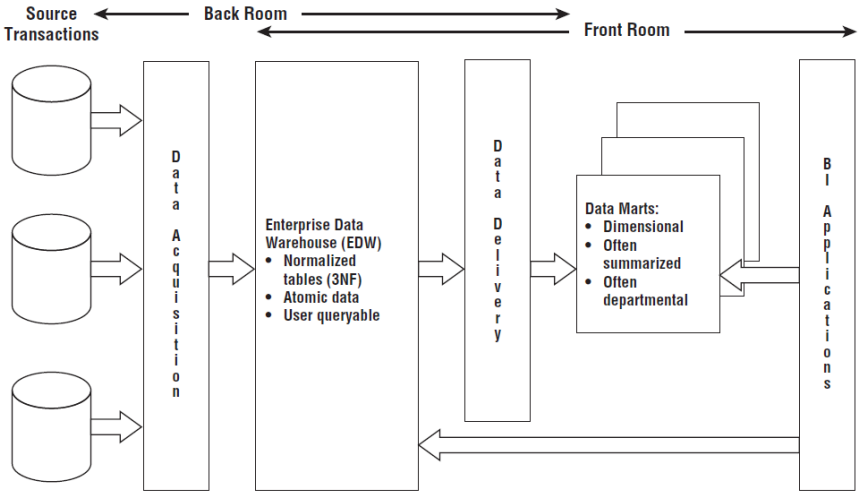
Vlastnosti a účel vrstev zůstává stejný i v jiných architekturách, pouze se liší jednotlivé požadavky na složení těchto vrstev – například v IDL datový model centrální databáze vytvořený podle 3NF nebo dimenzionálního modelování.

1.4 Základní architektury datových skladů

V této sekci se zaměřím na popis rozdílů mezi architekturami zakkladatelů datových skladů. Mezi architekturami Billa Inmona a Ralpha Kimballa je vidět jejich odlišný pohled na celý koncept datového skladu. Většina dalších architektur datových skladů vychází právě z jedné z těchto dvou, kterou většinou rozšiřuje nebo upravuje. Z tohoto důvodu můžeme označit tyto dvě architektury jako základní.

1.4.1 Datové sklady dle Inmona

Inmonova architektura se také nazývá Hub and Spoke nebo Corporate Information Factory (CIF). Celý způsob návrhu datového skladu Billem Inmonem se označuje jako top down přístup (odshora dolů) [18]. Tímto přístupem nejdříve kompletně analyzujeme všechny potřebné zdrojové systémy, vytvoříme návrh jednotného datového modelu a až nakonec vytváříme datová tržiště dle obchodních požadavků nad podmnožinou dat uložených v centrální databázi.



Obrázek 1.2: Architektura datového skladu dle Inmona [19]

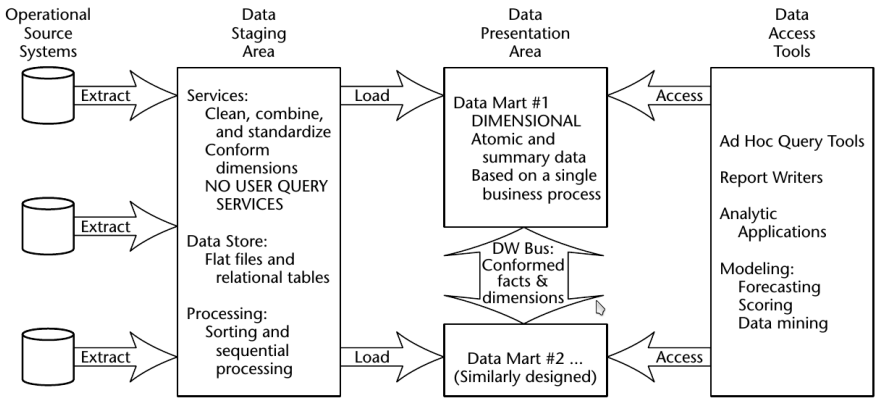
Na obrázku 1.2 je zobrazena architektura datového skladu dle Inmonna. Z obrázku lze vyčíst, že Inmonnův datový sklad a jeho jednotlivé vrstvy odpovídají současné architektuře datového skladu popsané v sekci 1.3. Jednotlivé vrstvy jsou pouze přejmenovány:

- Source Transaction = Source Layer,
- Data Acquisition = Data Integration Layer,
- Enterprise Data = Integrated Data Layer,
- Data Delivery = Access Layer,
- BI Applications = Information Delivery Layer.

Hlavní rozdíl se nachází ve vrstvě Data Delivery (Access Layer), protože v Inmonově architektuře není v této vrstvě začleněna sémantická databáze. Nachází se zde pouze jednotlivá datová tržiště. Z tohoto důvodu může v jednotlivých datových tržištích docházet k různé definici jednotlivých obchodních entit – mohou se lišit informace získané z datových tržišť s podobným obsahem, pokud mají jinak zadefinované klíčové obchodní entity (v této architektuře nemusí existovat „jednotná pravda“).

1.4.2 Datové sklady dle Kimballa

U Kimballovy architektury se můžeme setkat i s označením Data mart bus architecture. Oproti Inmonovi používá Kimball přístup návrhu datového skladu označovaný jako bottom up (odzdola nahoru). Tento přístup spočívá v získávání dat ze zdrojových systémů pouze pro potřeby naplnění datových tržišť specifikovaných dle obchodních požadavků (nedochází k ukládání nevyužívaných atributů a ke komplexní analýze zdrojových systémů).



Obrázek 1.3: Architektura datového skladu dle Kimballa [19]

Na obrázku 1.3 je zobrazena Kimballova architektura. Na první pohled je vidět odebrání vrstvy s integrovanými daty. V Kimballově architektuře slouží pro ukládání dat pouze jednotlivá datová tržiště – datový model celého skladu (všech datových tržišť) je navržen pouze pomocí dimenzionálního modelování.

Pokud tuto architekturu porovnáme s architekturou popsanou v sekci 1.3, tak zjistíme, že některé vrstvy jsou opět pouze přejmenované:

- Operation Source System = Source Layer,
- Data Staging Area = Data Integration Layer,
- Data Access Tools = Information Delivery Layer.

Hlavní rozdíl se nachází v pojetí Data Presentation Area. V Kimballově datovém skladu chybí centrální databáze a tedy chybí i celá IDL, jak ji známe z předchozí architektury. Tato centrální databáze je nahrazena jednotlivými datovými tržišti, kde každé datové tržiště implementuje právě jeden obchodní proces.

1.4.3 Shrnutí základních rozdílů

Hlavní rozdíly mezi Inmonovou a Kimballovou architekturou datového skladu jsou vypsány v tabulce 1.1 [20]. Z tabulky je patrné, že architektura dle Inmona je robustnější a lépe podporuje integraci mnoha systémů a snadněji udržuje správnost podnikových informací při změně obchodních pravidel. Na druhou stranu tato architektura vyžaduje vyšší počáteční investici. Oproti tomu architektura dle Kimballa je vhodnější pro menší instituce s ustálenými obchodními procesy, nebo pro instituce, které mají omezený rozpočet na vytvoření datového skladu.

Byl proveden výzkum, který analyzoval podle jakých kritérií instituce volí architekturu datového skladu. V tomto výzkumu bylo zjištěno, že instituce s jasně stanovenou informační strategií (zahrnuje aktivní využívání datového skladu v informační infrastruktuře), se zkušenými IT pracovníky a s většími finančními zdroji volí typ architektury dle Inmonna. Oproti tomu instituce s nižšími zdroji a s potřebami rychlého vytvoření datového skladu volí typ architektury dle Kimballa. [15]

1.5 Dimenzionální modelování

Dimenzionální modelování je specifický způsob návrhu databázového modelu, který je oproti klasickému návrhu databázového modelu používanému v běžných informačních systémech (3NF – třetí normální forma) mnohem intuitivnější pro koncové uživatele [19].

Normalizovaný model databáze je navržen tak, aby odstraňoval duplicitní data (kvůli tomu rozprostírá data do mnoha samostatných entit). Z tohoto

Inmonova architektura	**Kimballova architektura**
• Centralní databáze normalizovaná dle 3NF • Datový model centrální databáze je navržen nad celou institucí (top down přístup) • Neredundantní data (normalizovaná databáze) • Delší a rigidní implementace • Jednodušší integrace dalšího systému a změny obchodních požadavků	• Množina datových tržišť navržených dle dimenzionálního modelování • Datový model jednotlivých datových tržišť je navržen dle požadavků konkrétního oddělení (bottom up přístup) • Redundantní data (dimenzionální model) • Kratší a flexibilní implementace • Složitá integrace dalšího systému a změny obchodních požadavků

Tabulka 1.1: Základní rozdíly mezi Inmonovou a Kimballovou architekturou

důvodu je takto vytvořená databáze vhodná pro provozní informační systémy (rychlé pro operace insert/update/delete). Nicméně z důvodu velkého množství spojování jednotlivých entit, je tento model nevhodný pro analytické dotazy.

Oproti tomu dimenzionální model databáze je velice populární pro analytické dotazy. Tento model je základem pro zpracovávání analytických dat pomocí tzv. datových kostek – technologie OLAP (viz sekce 1.8), která umožňuje efektivní zpracování velkého množství dat. Dimenzionální modelování umožňuje ukládat i nadbytečná data (nesplňuje standardně používané databázové normalizace, ale razantně redukuje potřebné množství databázových spojení). Dimenzionální modely databáze se skládají pouze z faktových a dimenzionálních tabulek. [21]

Faktová tabulka je tabulka, ve které jsou umístěny metriky, které se používají k analytickým výpočtům. V této tabulce se nachází cizí klíče, které odkazují do jednotlivých dimenzionálních tabulek. Primárním klíčem faktových tabulek bývá výjimečně nový umělý klíč, častěji se používá n-tice všech cizích klíčů do dimenzí. Tyto tabulky obsahují velké množství záznamů (základní data pro analytické dotazy – metriky), oproti dimenzionálním tabulkám je tedy jejich velikost několikanásobně větší.

Dimenzionální tabulka rozšiřuje data obsažená ve faktových tabulkách o další informace (popisují detailněji jednotlivá fakta). Na základě těchto informací můžeme při tvorbě analýz vybrat, které záznamy z faktové tabulky

nás zajímají. Zjednodušeně tedy můžeme říct, že dimenzionální tabulka slouží jako filtr, který určuje, jaké řádky chceme vybrat z faktové tabulky.

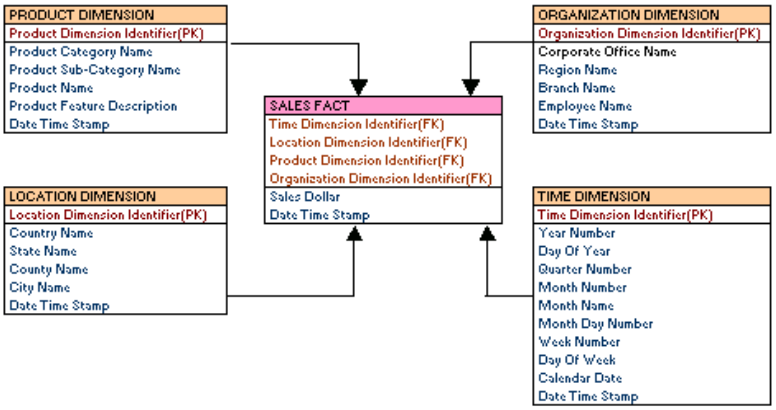
V dimenzionálním modelování existují dva základní typy způsobu návrhu schémat. Jsou to schémata typu star (hvězda) a snowflake (sněhová vločka).

Star schéma obsahuje pouze dimenzionální tabulky přímo navázány na faktovou tabulku. Tento druh schématu je nejjednodušší a dotazy nad ním obsahují nejmenší počet databázových spojení. Na druhou stranu v dimenzi dochází často k tvorbě duplicitních záznamů a tedy k větším požadavkům na kapacitu úložiště.

Snowflake schéma je obdoba star schématu, ale dimenze mohou být rozloženy do více tabulek. Tedy každá dimenze nemusí být navázána na faktovou tabulku. Tento způsob návrhu nám do určité míry umožňuje normalizovat data uložená v dimenzích (můžeme odstraňovat duplicitní záznamy dekompozicí), ale dotazy obsahují větší počet databázových spojeni. Proto je toto schéma sice paměťově výhodnější, ale zároveň výpočetně náročnější.

Na obrázcích 1.4 a 1.5 je vidět struktura jednotlivých typů schémat a jsou zde vidět i popisované rozdíly mezi těmito dvěma způsoby návrhu.

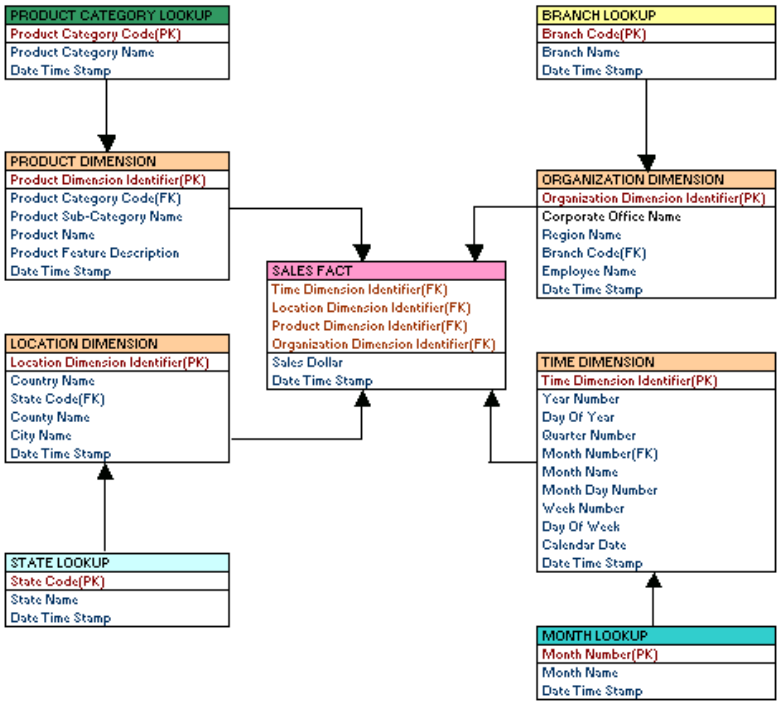
V praxi se používají ještě různé deriváty z těchto dvou základních typů schémat. Jsou to například schémata označovaná jako galaxie (více faktových tabulek ve star schématu) nebo snowstorm (sněhová bouře – více faktových tabulek v snowflake schématu).



Obrázek 1.4: Dimenzionální model – star schéma [22]

1.6 Slowly changing dimension

Slowly changing dimension (zkráceně SCD) si lze představit jako způsob ukládání historizovaných záznamů do datového skladu. Tento pojem představil



Obrázek 1.5: Dimenzionální model – snowflake schéma [23]

Ralph Kimball až v roce 1996, proto SCD bylo nejdříve definováno na dimenzích, tedy na dimenzionálních schématech (viz struktura datového skladu dle Kimballa v sekci 1.4.2) [24].

Tento koncept je navržen tak, aby se vypořádal s rozdílnými požadavky na způsob ukládání historických záznamů u jednotlivých tabulek. Ve svém základním znění rozlišuje tři základní způsoby historizace (SCD typu 0, 1, 2). Existují i další typy SCD (ve své knize Ralph Kimball již definuje sedm různých typů [19]), ale tyto další způsoby se používají pouze pro výjimečné situace, které mohou při návrhu datového skladu nastat – nejsou tak běžné jako první tři [19].

V případě návrhu datového skladu pomocí normalizovaného databázového schématu (například architektura dle Inmona, viz sekce 1.4.1) se používá nejčastěji SCD typu 2. Můžeme se i setkat s konceptem, kde každá tabulka má svoji aktuální tabulku (head table) a historickou tabulku (version table). Aktuální tabulka používá SCD typu 1 (přepisují se neaktuální záznamy) a historická tabulka používá SCD typu 2 [13] (neaktuální záznamy jsou ukládány s rozsahem jejich platnosti – začátek/konec platnosti).

Kromě klasické jednoosé historizace (při ukládání rozsahu platnosti se řídíme časovým otiskem, při kterém se záznam zpracovával datovým sklady) existuje i víceosá historizace. Víceosá historizace se řídi i jinými časovými

otisky. Většinou jsou tyto rozsahy platnosti určeny obchodními procesy organizace – například platnost smlouvy a podobně. Takto navržená historizace obsahuje více technických atributů (s informacemi o rozsazích platnosti jednotlivých záznamů), ale umožňuje lépe vyhovět požadavkům uživatelů při vytváření historických analýz.

1.6.1 SCD typu 0

Historizace SCD typu 0 se také nazývá držení originálu (anglicky retain original). Záznam vložený do tabulky není možné nijak měnit v čase.

1.6.2 SCD typu 1

Také označovaná jako přepisování (anglicky overwrite) historie. Při změně záznamu v tabulce dochází k jeho přepsání za aktuální hodnoty. Tento způsob historizace odstraňuje veškerou historii. V tabulkách 1.2 a 1.3 je simulováno chování tohoto způsobu historizace v případě změny počtu kreditů u předmětu.

Id předmětu	**Název předmětu**	**Kód**	**Kredity**
156512	Magisterská práce	MI-DIP	23

Tabulka 1.2: Ukázka chování SCD typu 1: před změnou záznamu

Id předmětu	**Název předmětu**	**Kód**	**Kredity**
156512	Magisterská práce	MI-DIP	30

Tabulka 1.3: Ukázka chování SCD typu 1: po změně záznamu

1.6.3 SCD typu 2

Historizace dle SCD typu 2 se vyznačuje přidáním nové řádky při změně záznamu. Tento způsob historizování záznamů v tabulce udržuje kompletní historii změn záznamu se stejným obchodním klíčem (anglicky business key, zkráceně BK, je jednoznačný identifikátor záznamu používaný ve zdrojových systémech nebo v rámci obchodních procesů společnosti).

Pro správné fungování tohoto typu historizace je nutné přidat do historizované tabulky další atributy. Nejčastěji se k tomuto účelu používají následující atributy (tyto atributy se mohou lišit v jednotlivých implementacích, ale vždy je nutné u tohoto typu SCD uchovávat rozsah platnosti záznamu).

- *Date to* (datový typ timestamp) označuje dokdy je záznam platný. V případě aktuálního záznamu se jedná o konstantu, která nám ukazuje ne-

konečnou platnost (v ukázkových tabulkách jsem použil konstantu 31. 12. 2999).

- *Date from* (datový typ timestamp) označuje odkdy je záznam platný. Při úvodním nahrání dat do datového skladu (iniciální nahrání datového skladu) je nastaven tento atribut na konstantu určující nekonečnou platnost v minulosti (v ukázkových tabulkách 1. 1. 1900), jinak je atribut nastaven na datum nahrání záznamu do datového skladu.
- *Version* (datový typ integer) určuje verzi konkrétního záznamu.
- *Technical key* (datový typ bigint – sekvence) je umělý klíč v tabulce, který je vytvářen pomocí sekvence.

Primární klíč historizované tabulky je nutné upravit, aby bylo možné přidat změněný záznam a uchovávat záznam historický. Pokud chceme využívat obchodní klíč (BK) ze zdrojového systému je nutné do n-tice primárního klíče přidat další atribut – nejčastěji volíme date_to nebo celý rozsah platnosti záznamu (atributy date_to a date_from).

Také můžeme využívat umělý klíč vytvářený pomocí sekvence. Takto vytvořený atribut nazýváme technickým klíčem (anglicky technical key, zkráceně TK).

Technický klíč nebo n-tice obchodního klíče s rozsahem platnosti záznamu jednoznačně identifikují právě jeden záznam v historizované tabulce a můžeme je tedy použít jako primární klíč tabulky.

V tabulkách 1.4 a 1.5 je znázorněno chování historizované tabulky při změně počtu kreditů v předmětu MI-DIP.

Id předmětu	Název předmětu	Kód	Kredity	tech_key	date_from	date_to	ver.
156512	Magisterská práce	MI-DIP	23	1235	1.1.1900	31.12.2999	1

Tabulka 1.4: Ukázka chování SCD typu 2: před změnou záznamu

Id předmětu	Název předmětu	Kód	Kredity	tech_key	date_from	date_to	ver.
156512	Magisterská práce	MI-DIP	23	1235	1.1.1900	4.2.2017	1
156512	Magisterská práce	MI-DIP	30	4582	4.2.2017	31.12.2999	2

Tabulka 1.5: Ukázka chování SCD typu 2: po změně záznamu

1.7 Získávání dat

Pro nahrávání dat ze zdrojových systémů do datového skladu slouží takzvané ETL (anglicky extract, transform, load) procesy. Někdy je nutné získaná data

nejdříve uložit, a pak je nad cílovou databází transformovat. Tento postup se nazývá ELT (extract, load, transform). Často je nutné použít kombinaci těchto dvou metod.

V ETL procesech dochází k extrakci dat ze zdrojových systémů, tedy z datového modelu konkrétního provozního nezávislého systému. Tato data je nutné transformovat, aby je bylo možné uložit do databáze popisující celou obchodní doménu organizace (datový model použitý ve vrstvě s integrovanými daty viz. sekce 1.3.3). Je nutné provést základní čištění především s ohledem na technickou kvalitu dat (primární klíče, cizí klíče), provést standardizaci dat (např. jednotné konstanty pro příznak ano/ne) a do cílové tabulky nahrát transformovaná data s ohledem na typ historizace používaný v cílové tabulce.

ETL procesy nám tedy zajišťují základní čistotu, standardizaci a integraci dat z nezávislých systémů pro potřeby datového skladu.

ETL procesy můžeme implementovat pomocí jazyka SQL a jeho procedurárního rozšíření (v případě databáze PostgreSQL se jedná o PL/pgSQL) nebo pomocí externích ETL nástrojů, které provádějí datové transformace mimo databázi. V případě komerčních nástrojů se jedná například o Data Integrator od společnosti Oracle, open-source nástroje zastupuje například Pentaho Data Integration.

1.8 Technologie OLAP

Online analytical processing (zkráceně OLAP) je způsob zpracování velkého množství dat, který sdílí tři základní charakteristiky. Těmito charakteristikami jsou [11]:

- *Analytické technologie pro multidimenzionální data* pomáhají vytvářet kvalitní analýzy nad multidimenzionálními daty. Pod tímto pojmem si můžeme představit pokročilé agregační funkce, statistické funkce a vizualizační techniky.

- *Pokročilá databázová podpora* umožňuje přístup k datům na různých typech databázových systémů, zajišťuje plynulou dobu odezvy pro jednotlivé dotazy a umožňuje využívat data i z velkých databází (řádově terabyty dat).

- *Uživatelský přívětivé rozhraní pro koncové uživatele* je pro OLAP technologie klíčové. Protože správně naimplementované uživatelské rozhraní umožňuje uživatelům urychlit rozhodování (v případě managementu) a vytváření analýz (v případě analytiků) nad vybranými daty.

Tato technologie slouží pro efektivní zpracování velkého množství dat. Často se s její pomocí zpracovávají data uložená v datových skladech (velké množství záznamů, které je potřeba v přijatelném čase zpracovat).

1.8.1 Architektura OLAP

Systémy OLAP mají tři základní architektonické komponenty. Tyto komponenty mohou být na stejném počítači nebo distribuovány mezi několik počítačů [11].

- *Graphical user interface* vytváří rozhraní pro vizualizaci dat.
- *Analytical processing logic* zpracovává data dle zadaných kritérií.
- *Data-processing logic* získává data ze zdrojových systémů.

Tato architektura je znázorněna na obrázku 1.6. OLAP systémy mohou získávat data přímo z operačních provozních systémů nebo z datových skladů. Většina těchto systémů umožňuje výsledná data exportovat do Microsoft Excelu, či jiných aplikací nebo data zobrazovat například pomocí reportů a dashboardů.

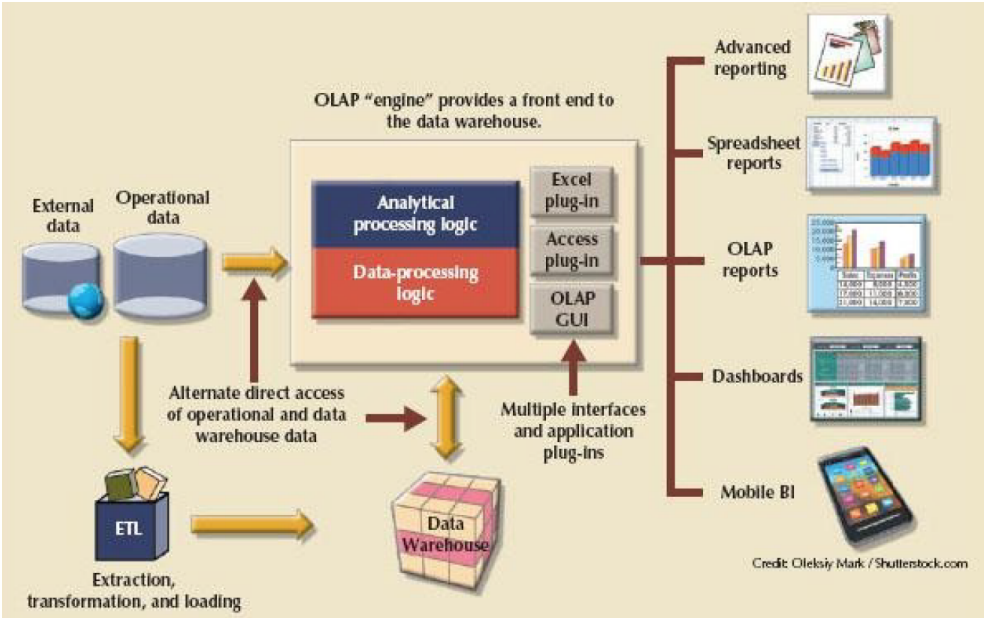
Datová kostka (anglicky data cube) je způsob organizace dat, který rozšiřuje klasické tabulky (dvou dimenzionální tabulky běžně používané v reálném životě) do prostoru, tedy je rozšiřuje o další dimenze. Tento způsob organizace a pohledu na data je využíván v technologii OLAP.

Pro analýzy nad datovou kostkou se používají speciální operace. Mezi nejčastější operace patří krájení (slice), kostkování (dice), pohyb v dimenzi k/od detailů (drill down/up). Pohyb v dimenzi od detailu si můžeme představit jako hrubší/jemnější pohled na informace (z měsíce se přesouváme k roku).

1.8.2 Typy systému OLAP

ROLAP (Relational OLAP) umožňuje využívat technologii OLAP na klasické relační databázi za podmínky, že zdrojová data jsou uložena v dimenzionálním schématu (viz dimenzionální modelování v sekci 1.5). Tento typ OLAP systému nevyžaduje předpočítávání a cachování dat uložených v databázi, protože operace používané nad datovou kostkou jsou překládány do jazyka SQL.

MOLAP (Multidimensional OLAP) transformuje zdrojová data do datových kostek, které ukládá do multidimenzionálních polí (zdrojová data jsou předpočítávána a cachována). Operace nad datovou kostkou probíhají přímo nad multidimenzionálními poli, ne nad databází. Tato vlastnost se často využívá při zpracovávání velkého množství dat, kde tímto odpadají problémy s odezvou databázových systémů při rozsáhlých dotazech.



Obrázek 1.6: Architektura technologie OLAP [11]

Kapitola 2

Jmenná konvence v databázových systémech

Jmenná konvence popisuje, co je nám známo o způsobu vytváření názvů a jmen.

První typ jmenné konvence se označuje jako konvence jednoduše popisná (simply descriptive). Registrační autorita nemá kontrolu nad způsobem vytváření jmen a pouze popisuje již existující jména.

Druhý typ jmenné konvence se nazývá předepisující (prescriptive). Tento typ konvence specifikuje registrační autoritou, jak by měla být jednotlivé jména formulována, aby byla v souladu s definovanou jmennou konvencí.

Jmenná konvence by měla být specifikována v dokumentu, který by měl obsahovat [25]:

- Popis domény,
- jména a názvy již definované registrační autoritou v instituci,
- sémantická pravidla,
- syntaktická pravidla,
- lexikální pravidla (délka, typ proměnných, jazyk),
- požadavky na unikátnost názvů.

2.1 Jmenná konvence a databáze

Hlavním cílem používání jmenných konvencí v databázových systémech je, že kdokoliv může snadno identifikovat účel a typ všech databázových objektů, které databáze obsahuje [26].

2.1.1 Výhody použití jmenné konvence v databázi

Výhody, které nám poskytuje použití jmenné konvence při vývoji a správě databáze, jsou následující [26]:

- Poskytuje jednotný standard pro pojmenování všech databázových objektů,
- redukuje úsilí nutné pro přečtení a porozumění kódu SQL, či jeho procedurální nástavbě,
- vylepšuje pochopení v případě potencionálních víceznačností,
- poskytuje rychlejší pochopení databázový objektů novými uživateli,
- řadí podobné databázové objekty do posloupnosti, která redukuje čas potřebný k nalezení specifického objektu v databázi.

2.1.2 Základní problémy při vytváření jmenné konvence v databázi

Plularita podstatných jmen v názvech databázových objektů je často diskutovaný problém spojený s databázovými jmennými konvencemi. Dříve se častěji používalo množné číslo pro názvy tabulek (protože každá tabulka obsahovala více záznamu – například tabulka obsahující informace o studentech obsahuje informace o více studentech, tudíž byla pojmenována jako Studenti).

Naopak v současnosti se preferuje používání jednotného čísla. Názvy v jednotném čísle lépe vystihují datový model – Student studuje Předmět. Také využívání jednotného čísla dává větší smysl v různých jazycích pro nerodilé mluvčí [27].

Způsob psaní velkých/malých znaků a oddělovačů slov (case convention) v názvech je další problém, který má spoustu možností řešení. U tohoto problému je nutné zohlednit omezení konkrétních databázových systémů (například Oracle ve výchozím nastavení konvertuje vše na velká písmena, PostgreSQL na malá písmena). Pro způsob zápisu oddělovačů slov v názvech máme následující možnosti. [26] [27]

- *Pascal Case*: ZapsanyPredmet, v případě počtů znaků menším než tři budou všechny znaky ve slově velké,
- *Camel Case*: zapsanyPredmet, v případě počtů znaků menším než tři budou všechny znaky ve slově velké,
- *Underscore*: zapsany_predmet nebo různé kombinace s velkými/malými počátečními písmeny).

V době psaní této práce je nejoblíbenější oddělovač slov typu Underscore (podtržítko), zejména pro databáze Oracle a PostgreSQL.

2.2 Tvorba jmenné konvence pro databáze

Při tvorbě jmenné konvence je vždy nutné zohlednit obchodní doménu, pro kterou budeme datový model databáze vytvářet a specifika databázového systému, který budeme používat.

Dle specifik databázového systému je nutné definovat maximální délku jednotlivých jmen a způsob použití velkých/malých písmen (letter case). Také se dle specifik databázového systému určuje jaké databázové objekty je nutné do notace zahrnout (například materializované pohledy nemusí být implementovány ve všech databázových systémech).

Znalost obchodní domény určí používaný jazyk (doporučuje se používat pouze jeden), hodnoty některých atributů (typicky příznakové atributy ano/ne) a případně další požadavky – např. další členění jmen tabulek z důvodu velkého rozsahu domény (velký počet tabulek s podobným významem do skupin).

Každá databázová jmenná konvence by měla obsahovat tyto základní požadavky [11]:

- Požadavky na jména tabulek
  - Krátká a výstižná slova spojená s obchodní doménou.
  - Dokumentovat zkratky, synonyma a přezdívky pro každou entitu.
  - Unikátnost jména v datovém modelu.
  - Použití jednotného nebo množného čísla (stejné pro všechny názvy).
  - Pro relační tabulku (dekompozice vztahu M:N) vytvořit jméno pomocí kombinace tabulek. spojených dekomponováním vztahu
- Požadavky na jména atributů
  - Unikátnost v rámci entity.
  - Správně popisovat charakteristiku atributu.
  - Přípony pro speciální typy atributů (např. _PK pro primární klíč).
  - Nejedná se o klíčové slovo konkrétní databáze.
  - Bez mezer a zvláštních znaků (@, &, atd.).
- Požadavky na jména vazeb
  - Sloveso popisující podstatu vazby.

Jmenná konvence pro další databázové objekty se definuje pomocí obdobných pravidel. Například požadavky na jména databázových pohledů (views) jsou stejné jako požadavky u tabulek, ale ještě přidáme před název předponu V_.

Pokud v databázi plánujeme používat i procedurální rozšíření jazyka SQL, měla by jmenná konvence zahrnovat i metodiku pro psaní zdrojového kódu (požadavky na strukturu kódu, názvy proměnných, názvy samotných funkcí a procedur).

2.3 Příklady v jazyce SQL

Na kódu SQL 2.3 vidíme příklad dotazu nad databází, která nepoužívá jmennou konvenci. Na tomto příkladě se můžeme setkat s následujícími problémy:

- Nejednotné pojmenovávání atributů,
- používání množného i jednotného čísla,
- nejednotný styl velkých/malých písmen (lowercase, camelcase...),
- špatně pojmenované a nejednotné primární a cizí klíče,
- nejednotné hodnoty příznakových atributů (ano/ne, true/false).

Všechny tyto problémy vedou k problematické práci s databází (jak pro analytiky tak pro administrátory), zejména pokud se jedná o databázi s velkým počtem tabulek. Prodlužují především délku tvorby jednotlivých analytických dotazů, délku potřebnou k seznámení s obsahem databáze novými pracovníky a zvětšují pravděpodobnost chyby při psaní analytických dotazů (databázové spojení podle jiných klíčů, špatně pochopené hodnoty v příznakovém atributu apod.).

Pokud tento příklad porovnáme s kódem SQL 2.3, který obsahuje dotaz nad databází vyvíjené za pomoci jmenné konvence, na první pohled vidíme rozdíl. Jmenná konvence i na takto jednoduchém dotazu zlepšuje jeho srozumitelnost a pochopitelnost (známe primární a cizí klíče, známe hodnoty příznakových atributů, jména atributů dávají větší smysl).

```
SELECT
 orgj.kod_jednotky,
 zam.prijmeni_zamestnance,
 zam.rodne_cislo,
 plat.soucasny_plat

FROM
 zamestanci_spolecnosti zam
 LEFT JOIN OrganizacniJednotky orgj
    ON zam.id_zamestnancovi_jednotky = orgj.orgj_id
 LEFT JOIN Platy_zamestnancu plat
    ON plat.osobni_cislo = zam.peridno_id

WHERE plat.del != '*' AND zam.pracuje = 'Ano';
```

SQL 2.1: Dotaz nad databází bez jmenné konvence

```
SELECT
 orgj.kod,
 zam.prijmeni,
 zam.rodne_cislo_id,
 plat.plat_aktualni

FROM
 t_zamestnanec zam
 LEFT JOIN t_organizacni_jednotka orgj
    ON zam.fk_organizacnijednotka = orgj.organizacni_jednotka_pk
 LEFT JOIN t_mzdy plat
    ON plat.fk_zamestnanec = zam.zamestnanec_pk

WHERE plat.smazany != '1' AND zam.pracuje = '1';
```

SQL 2.2: Dotaz nad databází se jmennou konvencí

Část II

Praktická část

Kapitola 3

Datový model datového skladu

V této kapitole popisuji všechny činnosti související s vytvářením datového modelu centrální databáze datového skladu.

Nejprve bylo nutné vytvořit jmennou konvenci pro jednotné vytváření databázových objektu v datovém modelu. Poté byla provedena analýza tří zdrojových systémů, které jsou do centrální databáze integrovány. Další sekce se věnuje samotné tvorbě datového modelu v modelovacím nástroji. Poslední sekce této kapitoly obsahuje způsob převodu datového logického modelu na fyzický datový model.

3.1 Jmenná konvence databázových objektů

Jmenná konvence byla vytvářena pro databázový systém PostgreSQL. Tato databáze má ve výchozím nastavení délku názvů databázových objektů (identifikátory) omezenou 63 znaky a všechny identifikátory, které jsou psány bez uvozovek, jsou psány ve výchozím nastavení databáze malými písmeny.

V budoucnosti možný přechod na databázi Oracle, která má identifikátory omezeny 30 znaky (ve verzi 12c je možné rozšířit až na 128 znaků) a všechny identifikátory jsou ve výchozím nastavení databáze psány velkými písmeny.

Z tohoto důvodu jsou všechny názvy databázových objektů psány velkými písmeny, v jednotném čísle a v českém jazyce bez diakritiky. Výjimku tvoří pouze databázové procedury a funkce, které jsou psané a pojmenované v angličtině z důvodu přehlednosti zdrojového kódu.

Kompletní znění mnou navržené databázové notace, které se používá v projektu datového skladu ČVUT se nachází v příloze B. V tomto přiloženém dokumentu se nachází:

- Způsob pojmenování databázových objektů (tabulky, pohledy, cizí tabulky atd.),
- způsob pojmenování speciálních atributů (primární klíč, cizí klíč),

- metodika pojmenovávání jednotlivých databázových vrstev,
- konstanty použité v datovém skladu,
- způsob tvorby asociací a dekompozice vztahu M:N.

3.1.1 Část jmenné konvence

V této sekci a jejích podsekcích se nachází část jmenné konvence, kterou jsem vytvořil pro pojmenování databázových objektů v datovém skladu. Celé znění notace je v příloze B.

Věnuji se zde metodice pojmenovávání dvou nejdůležitějších objektů – tabulek a atributů.

3.1.1.1 Názvy tabulek

Název každé tabulky je definován jednotným číslem velkými tiskacími písmeny. Jednotlivá slova jsou oddělena podtržítkem. Struktura tvorby názvu tabulky je následující:

[předpony]_[doména]_NAZEV_TABULKY_[přípony]

U jednotlivých databázových objektů, pomocí kterých jsou strukturována uložená data (objekty tabulkového typu), využíváme následující předpony:

- T – klasická tabulka,
- MV – materializovaný pohled,
- V – pohled,
- FT – cizí tabulka.

Domény, které používáme v názvu tabulek uložených ve schématu DWH (schéma s centrální databází) jsou definovány v sekci 3.3.1.1. Jedná se vždy o čtyřpísmenné zkratky domény, které usnadňují orientaci v centrální databázi podle základních datových entit.

Přípony používáme v názvech tabulek následující:

- DIM – dimenzionální tabulka dimenzionálního schématu.
- FACT – faktová tabulka dimenzionálního schématu.
- REL – relační tabulka (tabulka, která vzniká dekompozicí asociace typu M:N).
- NOHIST – nehistorizovaná tabulka.

Jednotlivé předpony i přípony mohou být skládány libovolně za sebou (vždy oddělené podtržítky), aby bylo možné správně definovat vlastnosti tabulek. Všechny názvy tabulek v databázi musí být unikátní i bez zkratek domén – tyto zkratky pouze zpřehledňují orientaci v centrální databázi, neslouží k odlišení jednotlivých tabulek.

3.1.1.2 Názvy atributů

Názvy atributů jsou opět definovány jednotným číslem a velkými tiskacími písmeny, kde jsou jednotlivá slova jsou oddělena podtržítkem. Struktura tvorby názvu atributu je následující:

[předpony]__NAZEV__ATRIBUTU__[přípony]

Nad atributy jsem definoval pouze jednu předpona, kterou je předpona FK. Tato předpona identifikuje atribut jako cizí klíč do tabulky. Dále jsou nad atributy definovány následující přípony:

- BK – business key, nebo-li obchodní klíč používaný ve zdrojovém systému (primární klíč zdrojové tabulky.
- TK – technický klíč, umělý klíč vygenerovaný v rámci sekvence.
- ID – jednoznačný identifikátor, který identifikuje řádek v tabulce (pokud neuvažujeme historizaci záznamů).

Nad všemi atributy s jednou z těchto přípon je vytvořeno integritní omezení NOT NULL. V případě obchodního klíče je název atributu určen názvem tabulky, ve které se nachází. U cizího klíče je název atributu určen názvem tabulky do které odkazuje (z názvu tabulky se vynechávají podtržítka reprezentující mezery).

3.1.2 Dotaz s využitím jmenné konvence

V dotazu SQL 3.1.2 je vidět použití této notace nad centrální databází datového skladu. Dotaz na získání záznamů o zapsaných předmětech studentů FIT je přehledný a snadno pochopitelný.

3.2 Zdrojové systémy

V současnosti jsou do datového skladu integrovány tři systémy: KOS, Anketa ČVUT a Závěrečné práce (FIT). Do budoucna plánujeme do datového skladu přidat studijní systémy fakulty FIT (Progtest, Edux – část s průběžným hodnocením studentů ze semestru, Marast) a celouniverzitní systém Usermap.

```
SELECT *
FROM dwh.t_zapi_zapsany_predmet
WHERE fk_studium IN (
  SELECT studium_bk
  FROM dwh.t_stud_studium
  WHERE fk_studijniprogram IN (SELECT studijni_program_bk
                               FROM dwh.t_stpl_studijni_program
                               WHERE fk_organizacnijednotka = 10));
```

SQL 3.1: Dotaz nad centrální databází datového skladu

3.2.1 Studijní systém KOS

Studijní informační systém KOS slouží k tvorbě rozvrhu, k zapisování studenta na akce a události spojené se studiem a k ukládání studijních výsledků.

Tento systém je používán všemi fakultami ČVUT. Některé fakulty nepoužívají všechny moduly studijního systému (například Fakulta jaderná a fyzikálně inženýrská nepoužívá modul pro tvorbu rozvrhu).

Tento systém je hlavním datovým zdrojem námi implementovaného datového skladu. Data získáváme z testovací databáze studijního systému KOS. Testovací databáze bývá jednou týdně přehrávána daty z produkční databáze.

Z tohoto systému zpracováváme čtyřicet šest tabulek, které obsahují informace o následujících datových entitách:

- Předměty a paralelky,
- studenti,
- učitelé,
- klasifikace a zkoušky,
- státní závěrečné zkoušky a závěrečné práce,
- studijní plány (tzv. bílá kniha),
- přihlášky ke studiu.

3.2.2 Systém Anketa ČVUT

Tento systém slouží k hodnocení předmětů a vyučujících studenty. Studenti tato hodnocení vyplňují vždy na konci semestru (zkouškového období).

Datové exporty jsou získávány pomocí souborů typu csv vždy pouze jednou pro každý běh ankety a to po jejím uzavření. Získáváme pouze tabulky spojené s hodnocením vyučujících a předmětů. Veškeré údaje o studentech, studijních výsledcích, vyučujících a předmětech nejsou z tohoto systému získávány, protože tyto data získáváme ze studijního systému KOS.

Celkem z tohoto systému zpracováváme osm tabulek, které obsahují textové, číselné hodnocení a informace spojené s jednotlivými anketami (oddíly, otázky, vypsané ankety apod.).

3.2.3 Systém Závěrečné práce

Systém Závěrečné práce je využíván pouze Fakultou informačních technologií. Tento systém spravuje závěrečné práce studentů zmíněné fakulty.

Slouží ke správě vypsaných témat závěrečných prací, k vybrání závěrečné práce studentem (s celým schvalovacím procesem), k přiřazení oponenta závěrečné práce a k ukládání posudků vedoucího a oponenta dle jednotné šablony Fakulty informačních technologií.

Všechna data spojená se závěrečnou prací jsou importována do systému KOS kromě kompletního znění posudků. Proto data spojená se závěrečnými pracemi získáváme pouze ze systému KOS a ze systému Závěrečné práce získáváme pouze posudky – ukládáme data pouze ze čtyř tabulek tohoto systému.

3.3 Tvorba datového modelu

Při tvorbě datového modelu centrální databáze datového skladu jsem postupoval ve třech krocích. V každém kroku bylo nutné vytvořit následující modely s odlišnou mírou detailu:

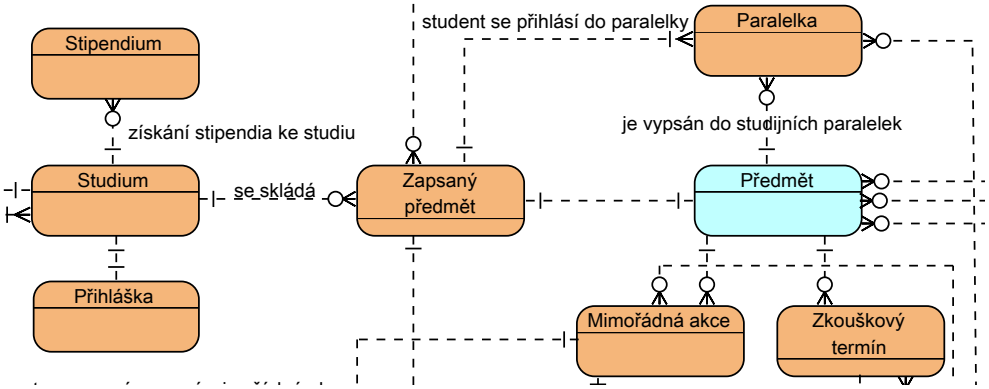
1. Entitní model domény studium na ČVUT.
2. Obchodní model centrální databáze.
3. Fyzický model centrální databáze.

3.3.1 Entitní model domény studium na ČVUT

Vývoj tohoto modelu nebyl ovlivněn datovými strukturami obsaženými ve zdrojových systémech. Primárním zdrojem informací pro tvorbu tohoto modelu byly obchodní procesy a předpisy platné na ČVUT.

Hlavním účelem tohoto modelu bylo správně definovat jednotlivé entity na ČVUT a vazby, které je spojují. Část takto vytvořeného modelu můžeme vidět na obrázku 3.1. Na obrázku lze vidět, že v tomto kroku tvorby datového modelu nás zajímají pouze hlavní entity a vazby mezi nimi (neřešíme jednotlivé atributy, databázové omezení apod.) – jedná se pouze o model hlavních entit v organizaci.

Z vytvořeného entitního modelu jsem vybral klíčové entity. Některé klíčové entity vznikly sloučením více entit obsažených v modelu. Klíčové entity nám poslouží jako základní datové entity, ze kterých budeme vytvářet tabulky při tvorbě datového modelu datového skladu.



Obrázek 3.1: Část entitního modelu v doméně studium na ČVUT

3.3.1.1 Základní datové entity

Ke každé základní datové entitě byla vytvořena její zkratka dle jmenné konvence (sekce 3.1). Z entitního modelu jsem vybral následující entity:

- AKCE Akce,
- ANKE Anketa ČVUT,
- AROK Akademický rok,
- KOUD Kontaktní údaje,
- MIST Místnost,
- ORGJ Organizační jednotka,
- OSOB Osoba,
- PARA Paralelka,
- PRED Předmět,
- PRIH Přihláška,
- STPL Studijní plán,
- STUD Studium,
- SZZK Státní závěrečné zkouška,
- UCIT Učitel,
- ZAPI Zapsaný předmět.

3.3.2 Obchodní model centrální databáze

Pod tímto označením se skrývá klasický fyzický model databáze. Ale v tomto modelu chybí technické atributy potřebné pro správné fungování datového skladu – zejména pro možnost historizování záznamů.

Takto vytvořený model by se mohl použít jako klasická OLTP[5] pro centrální informační systém nad celým ČVUT (obsahuje všechny entity získané z různých zdrojových systémů v integrované podobě). Tento databázový model, vytvořený v nástroji Enterprise Architect obsahuje:

- Tabulky rozdělené do jednotlivých základních datových entit,
- sloupce tabulek a definované obchodní primární klíče tabulek (většinou primární klíč ze zdrojového systému),
- vazby mezi tabulkami,
- databázové indexy,
- databázová omezení,
- dekompozice relačního schématu (vazby M:N, normalizace databázového modelu).

3.3.2.1 Mapování dat

Základní datové entity bylo v tomto kroku potřeba více rozpracovat již na základě dat, která můžeme získat ze zdrojových systémů. Zároveň při tvorbě databázového modelu bylo nutné splňovat metodiku určenou jmennou konvencí.

Byla provedena analýza, která určila důležité atributy pro jednotlivé základní datové entity obsažené ve zdrojových systémech. Informace o vybraných atributech ze zdrojových systémů jsme ukládali do datové logické mapy [6].

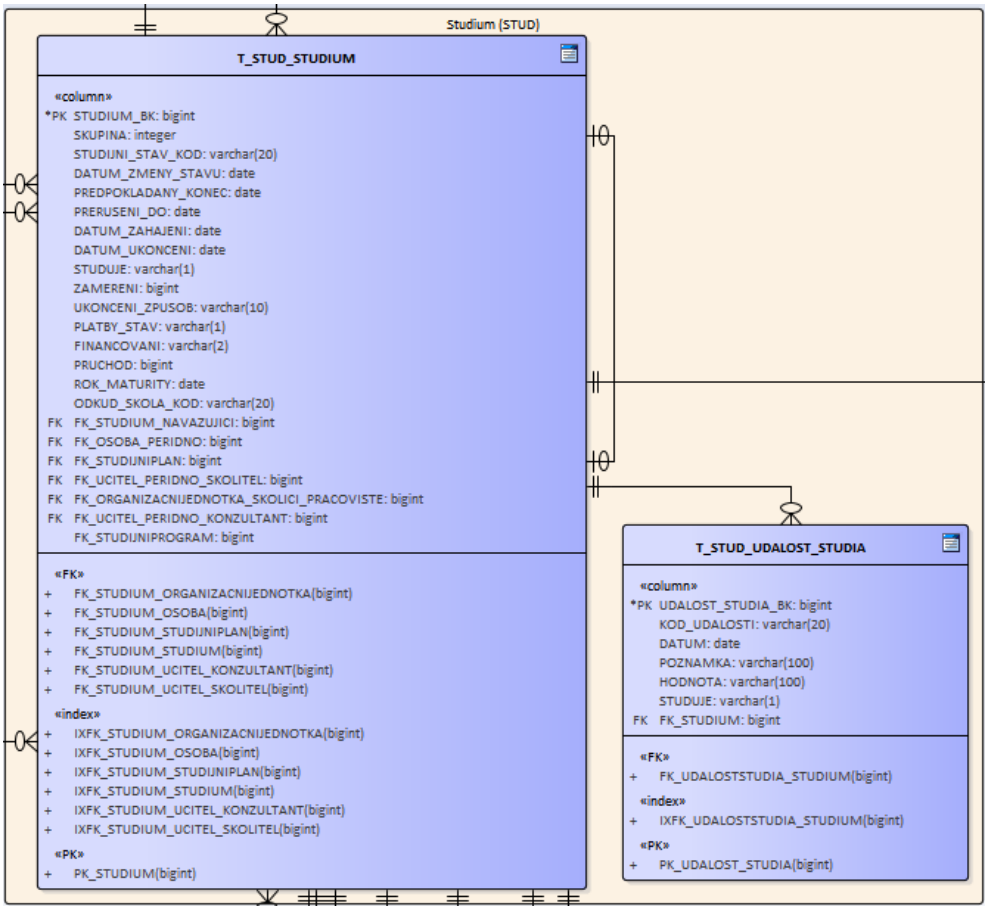
Následně byly atributy pro jednotlivé základní datové entity rozděleny do jednotlivých tabulek, aby splňovaly jak správný logický koncept, tak normalizaci dle třetí normální formy. Toto rozdělení v centrální databázi bylo také zaneseno do datové logické mapy. Tímto krokem nám vzniklo v datové logické mapě mapování atributů ze zdrojových systémů do centrální databáze datového skladu.

Tento způsob modelování obchodního modelu datového skladu lze vidět na obrázku 3.2, kde je zobrazena výsledná struktura entity Studium v nástroji

[5] Online Transaction Processing je technologie pro ukládání dat v databázi, která ji využívána v provozních systémech institucí s mnoha uživateli.

[6] Datová logická mapa je tabulka, která mapuje atributy ze zdrojových systémů do centrální databáze s případným popisem abnormálního chování (datová kvalita, složitější transformace apod.). Obsahuje technické a obchodní informace o zdrojovém systému, tabulce, atributu a o cílovém systému (v našem případě centrální databázi) tabulce, atributu.

Enterprise Architect po rozpracování. Celkově bylo namapováno 817 různých atributů ze zdrojových systémů do 73 tabulek umístěných v centrální databázi datového skladu.



Obrázek 3.2: Obchodní model entity Studium

3.3.3 Fyzický model centrální databáze

Podle fyzického modelu centrální databáze jsou definovány jednotlivé datové struktury uložené na databázovém systému. Jedná se o obchodní model centrální databáze, který je upraven, aby splňoval všechny požadavky kladené na datový sklad.

Z důvodu historizace je především nutné změnit obchodní primární klíče na nové technické primární klíče, které jsou vytvářeny sekvencí. Kdybychom neprovedli tuto změnu, nebylo by možné ukládat historické hodnoty záznamů, protože záznamu zůstává stejný obchodní klíč, pouze se mění hodnoty ostatních atributů. Abychom nenarazili při historizování záznamů na databázové

omezení unikátnosti primárního klíče je nutné do tabulek přidat nový unikátní technický primární klíč.

Také je obchodní model centrální databáze zbaven veškerých vazeb mezi tabulkami a databázových omezení spojených s cizími klíči. Dále je rozšířen o následující technické atributy:

- VERSION – verze záznamu (datový typ BIGINT).
- DATE_FROM – platnost záznamu od (datový typ TIMESTAMP).
- DATE_TO – platnost záznamu do (datový typ TIMESTAMP).
- LAST_UPDATE – datum poslední změny (datový typ TIMESTAMP).

V centrální databázi existují tabulky s názvy, které mají příponu „_NOHIST“. Těmto tabulkám nejsou přidány technické atributy, protože z vlastností dat obsažených v těchto tabulkách plyne, že nemohou být historizovány. Přidání technických atributů by bylo tedy bezpředmětné.

3.3.3.1 Převod obchodního modelu centrální databáze na fyzický

Převod obchodního modelu centrální databáze na fyzický probíhá přímo na databázovém systému. Nejprve je nutné vygenerovat z obchodního modelu SQL kód s DDL (data definition language). Tento kód spustíme na databázovém systému, tímto vzniknou datové struktury definované pomocí obchodního modelu.

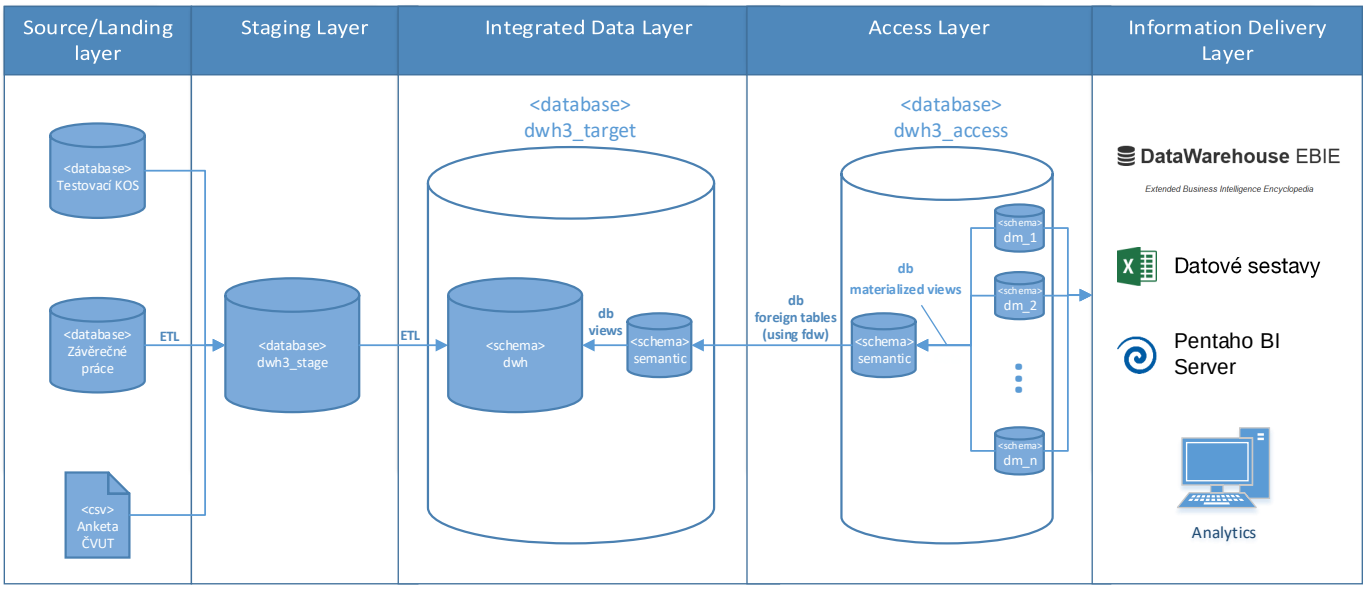
K samotné transformaci na fyzický model centrální databáze využíváme databázovou proceduru FC_GENERATE_TECHNICAL_COLUMNS, kterou jsem za tímto účelem vytvořil (její definice je v příloze C). Tato procedura změní primární klíče tabulek, vytvoří technické atributy, odstraní vazby mezi tabulkami, omezení spojená s cizími klíči a vytvoří indexy nad tabulkami.

Kapitola 4

Architektura datového skladu

V této kapitole se zaměřuji na implementaci jednotlivých částí architektury. Staging Layer a ETL procesy vysvětluji velice stručně, protože je detailně popisuje ve své diplomové práci Robert Kotlář [1].

Architektura datového skladu ČVUT verze námi označené jako DWH3[7] vychází ze současné architektury datových skladů popsané v teoretické části (viz sekce 1.3). Pozměněná architektura z teoretické části, která je využívána v datovém skladu verze DWH3 je znázorněna na obrázku 4.1.



Obrázek 4.1: Architektura datového skladu DWH3

[7] DWH3 je označení třetí generace prototypu datového skladu ČVUT implementovaného v rámci rozvojového projektu.

4.1 Source/Landing Layer

V případě systému KOS a systému Závěrečné práce nahráváme data přímo ze zdrojové databáze do Staging Layer. U systému KOS používáme jeho testovací databází běžící na databázovém systému Oracle 11g, u systému Závěrečné práce využíváme produkční databázi, která běží nad databázovým systémem PostgreSQL verze 9.5.1.

Systém Anketa ČVUT je nahráván do datového skladu pouze jednou za semestr. Proto jsme zvolili možnost získávání datových exportů formou CSV, které nahráváme do Staging Layer.

4.2 Staging Layer

Je tvořena datábází dwh3_stage běžící na databázovém systému PostgreSQL. Na této databází jsou vytvořeny následující typy schémat pro jednotlivé zdrojové systémy:

- *PRE_STAGE* – schéma obsahuje datový otisk tabulek ze zdrojového systému.
- *PRE_STAGE_CLEAN* – schéma obsahuje technicky vyčištěné tabulky ze zdrojových systémů.
- *PRE_STAGE_CLEAN_UDE* – schéma obsahuje technicky vyčištěné tabulky pro uměle definované entity[8].
- *STAGE_INCREMENT* – schéma, které obsahuje poslední otisk zdrojového systému, nahraný do datového skladu (obsahuje tabulky z PRE_STAGE i PRE_STAGE_CLEAN).
- *STAGE_INCREMENT_UDE* – speciální typ STAGE_INCREMENT pouze pro tabulky z PRE_STAGE_CLEAN_UDE.

Detailní význam tohoto členění a způsob zpracovávání dat pomocí Staging Layer je vysvětlen v diplomové práci Roberta Kotláře, který tuto vrstvu v datovém skladu verze DWH3 implementoval [1].

4.3 Integrated Data Layer

IDL je tvořena pomocí databáze dwh3_target, která je umístěna na databázovém systému PostgreSQL. Na této databázi jsou uložena dvě hlavní schémata – schéma dwh (reprezentuje centrální databázi) a schéma semantic (reprezentuje sémantickou databázi).

[8]Uměle definovaná entita je taková entita, která se nevyskytuje přímo v žádném zdrojovém systému, ale je definována pomocí sloučení více tabulek ze zdrojových systémů.

4.3.1 Centrální databáze

V centrální databázi (schéma dwh) jsou uloženy datové struktury definované fyzickým datovým modelem, který je popsán v kapitole 3. V této centrální databázi se v současnosti nachází 82 tabulek, které obsahují 1212 atributů (je zde momentálně nekonzistence v počtu tabulek vůči fyzickému datovému modelu, která je zapříčiněna přidáním tabulek pro vývoj a správu tohoto prototypu).

Data do centrální databáze jsou nahrávána pouze pomocí ETL procesů ze Staging Layer. Databáze obsažené ve Staging Layer a IDL nejsou spolu nijak propojeny.

4.3.2 Sémantická databáze

Sémantická databáze (schéma semantic) je v této architektuře tvořena databázovými pohledy nad centrální databází. Každý pohled definuje určitou obchodní entitu nad tabulkami z centrální databáze.

Je důležité, aby k jednotlivým databázovým pohledům obsažených ve schématu semantic měl uživatel semantic_agent oprávnění na použití příkazu SELECT. Bez tohoto oprávnění nebude správně fungovat propojení databází popsané v sekci 4.4.1.

4.4 Access Layer

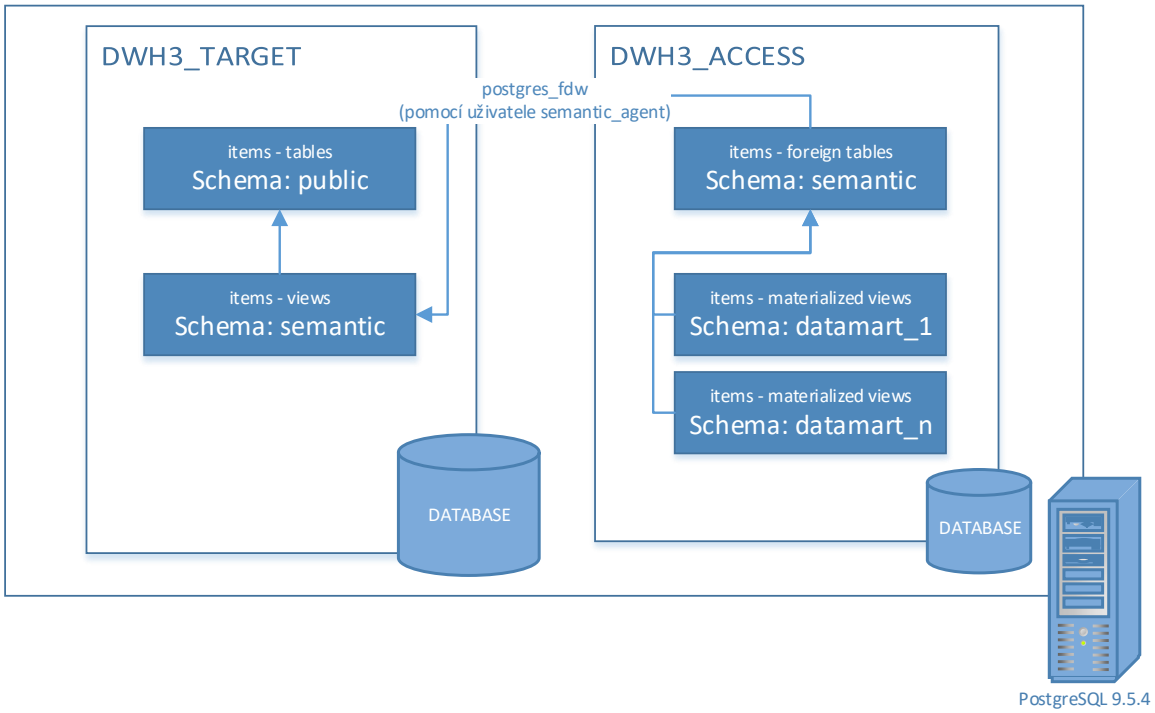
Access Layer je tvořena databází dwh3_access, která běží na databázovém systému PostgresSQL. Tato databáze slouží k ukládání jednotlivých datových tržišť a také k přístupu většího množství uživatelů (je nutné rozlišovat oprávnění uživatelů k přístupu pro jednotlivá datová tržiště).

4.4.1 Propojení se sémantickou databází

Databázové propojení mezi databázemi dwh3_access a dwh3_target je zajištěno pomocí technologie Postgres Foreign Data Wrapper [28]. Obrázek 4.2 znázorňuje způsob tohoto databázového propojení.

Pomocí uživatele semantic_agent a technologie postgres_fdw jsou propojeny databázové pohledy ze schématu semantic databáze dwh3_target do schématu semantic v databázi dwh3_access (příkaz IMPORT FOREIGN SCHEMA). Tyto databázové objekty jsou poté ve schématu semantic databáze dwh3_access reprezentovány jako cizí tabulky (foreign tables).

Toto propojení zajišťuje vetší bezpečnost dat obsažených v IDL, protože většina uživatelů datového skladu k datům přistupuje pomocí Acces Layer – nepřipojují se tedy v této architektuře k databázi, která obsahuje i centrální databázi se všemi daty.



Obrázek 4.2: Schéma propojení databází

4.4.2 Datová tržiště

Každé datové tržiště je uloženo do samostatného schématu. V tomto schématu jsou data reprezentována pomocí materializovaných pohledů, které jsou vytvářeny nad cizími tabulkami ze schématu semantic. Tímto se zajistí propojení pohledů až k datům, nad kterými jsou definována obchodní omezení a jsou uložena v centrální databázi umístěné v IDL.

Využití materializovaných pohledů má positivní dopad na dobu odezvy při získávání dat z datových tržišť. Nad těmito materializovanými pohledy jsou vytvořeny databázové indexy především na cizích klíčích, či na často používaných atributech.

4.5 Information Delivery Layer

Data uživatelům momentálně distribuujeme pomocí datových exportů v Excelu. Ve vývoji je obchodní encyklopedie EBIE a využíváme i Pentaho BI Server, který umožňuje analýzy nad datovými tržišti pomocí technologie OLAP (konkrétně se jedná o implementaci ROLAPU s názvem Mondrian).

4.5.1 DataWarehouse EBIE

Tato obchodní encyklopedie je také vyvíjena v rámci rozvojového projektu. Popisuje obchodní význam dat uložených v datovém skladu.

V tomto systému jsou dokumentovány jednotlivé obchodní procesy a entity. Navíc tato encyklopedie slouží také jako platforma pro poskytování reportů a dashboardů koncovým uživatelům datového skladu nebo pro žádosti související s poskytováním datových exportů uživatelům.

4.6 Použité technologie

V architektuře jsou využívány především dvě základní technologie – databázový systém PostgreSQL a ETL nástroj Pentaho Data Integration. Také jsou vývojáři využívány podpůrné nástroje pro správu databází (pgAdmin3, DataGrip) a pro tvorbu datových modelů (Visual Paradigm a Enterprise Architect).

4.6.1 PostgreSQL

PostgreSQL je objektově-relační databázový systém, který je vydávám jako open source software. Tento systém splňuje podmínky ACID a snaží se o implementaci standardu ANSI SQL. Je distribuován s procedurální nástavbou PL/pgSQL (obdoba PL/SQL používaného v databázích Oracle).

Všechny databáze zmíněné v architektuře běží na databázovém systému PostgreSQL verze 9.5.4. Pro správu databáze využíváme nástroj pgAdmin3 a pro tvorbu funkcí nebo složitějších SQL dotazů využíváme nástroj JetBrains DataGrip.

4.6.2 Pentaho Data Integration

Je multiplatformní open source ETL nástroj od společnosti Pentaho, který umožňuje tvorbu a spouštění ETL pomocí grafického rozhraní. Vytvořené transformace a úlohy je možné spouštět i pomocí příkazové řádky.

V tomto projektu využíváme šestou generaci tohoto nástroje pro tvorbu ETL. Pro spouštění ETL úloh využíváme možnost spouštění přes příkazovou řádku.

Kapitola 5

Implementace datových tržišť

Tato kapitola popisuje vývoj datových tržišť. Demonstruji v ní způsoby, jak jsou data poskytnutá z datového skladu v praxi využívána.

Nejprve popisuji vytváření obchodních entit v sémantické databázi. Dále se věnuji vytváření datových tržišť s počty studií a výsledky studentů.

V druhé části kapitoly se věnuji způsobům, které v současnosti používáme pro přístup k datům obsažených v datových tržištích. Jedná se o datové analýzy pomocí technologie ROLAP v nástroji Pentaho BI Server, interaktivní reporty v EBIE a klasický způsob vytváření datových sestav pomocí dotazů v jazyce SQL. Všechny tyto způsoby jsou využívány pro přístup k datům uživateli datového skladu ČVUT.

5.1 Definice obchodních entit v sémantické databázi

K vytvoření jednotlivých bloků (databázový objekt, který je definován podle obchodních pravidel) obsažených v sémantické databázi bylo potřeba určit klíčové obchodní entity domény studium na ČVUT. Ke splnění aktuálních požadavků na datové analýzy jsem stanovil tyto obchodní entity:

- Fakulta,
- katedra,
- student,
- studium,
- učitel,
- předmět,
- semestr,

- zapsaný předmět.

V databázi dwh3_target ve schématu semantic jsem vytvořil pro každou obchodní entitu pohled, který definuje její obchodní omezení a pravidla nad daty uloženými v centrální databázi datového skladu pomocí jazyka SQL.

Ukázku obchodních entit demonstruji v SQL kódu 5.1 na vytváření obchodní entity fakulta. V tomto kódu definuji fakultu nad tabulkou t_orgj_organizacni_jednotka z centrální databáze datového skladu. Pro tuto definici využívám pravidlo, že příznak jednotky je v případě fakulty ohodnocen hodnotou 102. Omezením nad atributem date_to získávám pouze aktuální záznamy z tabulky. Ostatní obchodní entity jsou vytvořeny obdobně.

```
1 CREATE VIEW semantic.v_fakulta AS
2   SELECT *
3   FROM dwh.t_orgj_organizacni_jednotka
4   WHERE priznak_jednotky = 102 AND date_to = '2199-12-31';
5
6 ALTER TABLE semantic.v_fakulta OWNER TO big;
7 GRANT SELECT ON TABLE semantic.v_fakulta TO semantic_agent;
```

SQL 5.1: Vytváření obchodní entity fakulta

5.2 Tvorba datových tržišť

Na základě požadavků na reporty jsem navrhl dvě datové tržiště. První obsahuje informace spojené s počtem aktivních studií v semestrech a druhé informace spojené s výsledky studentů v předmětech.

Obě datová tržiště se nacházejí v databázi dwh3_access a jsou umístěna v samostatných databázových schématech. Jejich model je tvořen pomocí dimenzionálního modelu, kde faktové i dimenzionální tabulky jsou reprezentovány pomocí databázových materializovaných pohledů.

Příklad vytvoření dimenzionální tabulky demonstruji na dimenzi fakulta. Tato dimenze je vytvořena pomocí SQL kódu 5.2. Jedná se o materializovaný pohled, který je vytvářen nad pohledem v_fakulta nacházejícím se v sémantické databázi umístěné na databázi dwh3_target. Tento pohled je přístupný pomocí databázového linku (viz sekce 4.4.1). Jelikož se nacházíme v Access Layer je nutné zpřístupnit vytvořený materializovaný pohled koncovým uživatelům. V tomto případě se jedná o databázového uživatele ebie.

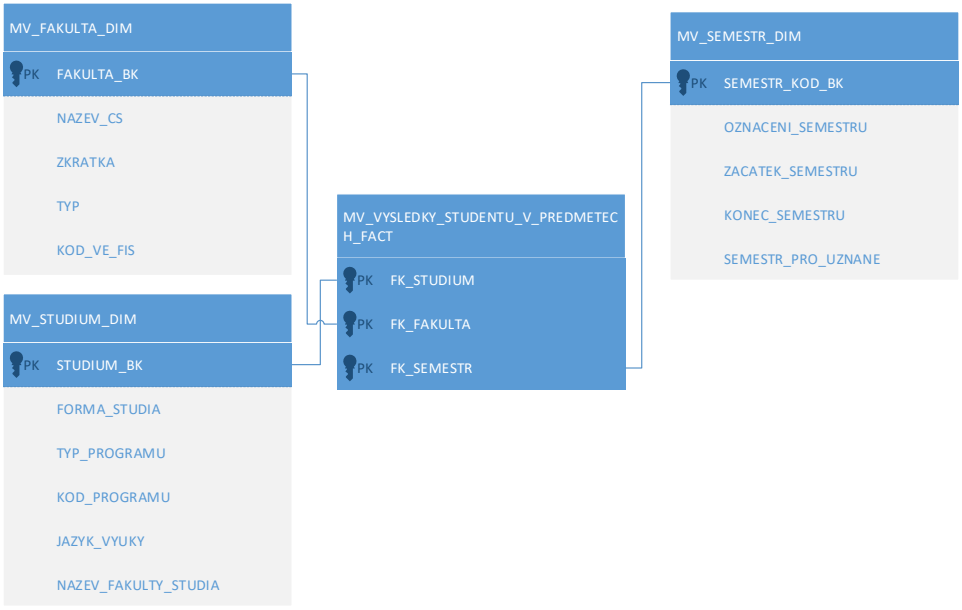
Všechny dimenze a faktové tabulky jsou vytvořeny obdobným způsobem. Pouze dochází ke změně zdrojových pohledů obsažených v sémantické databázi. Někdy je možné definovat tyto databázové objekty složitěji – spojením více pohledů v sémantické databázi nebo výpočtem metrik v případě faktových tabulek.

```
1  CREATE MATERIALIZED VIEW dm_vysledky_studentu_v_predmetech.mv_fakulta_dim AS
2    SELECT
3      organizacni_jednotka_bk AS fakulta_bk,
4      nazev_cs,
5      zkratka_cs,
6      priznak_jednotky        AS typ,
7      cislo_utvaru_id         AS kod_ve_fis
8    FROM semantic.v_fakulta WITH DATA;
9
10 ALTER TABLE dm_vysledky_studentu_v_predmetech.mv_fakulta_dim
11   OWNER TO big;
12 GRANT SELECT ON dm_vysledky_studentu_v_predmetech.mv_fakulta_dim TO ebie;
```

SQL 5.2: Vytváření dimenze fakulta

5.2.1 Datové tržiště s počtem studií v semestrech

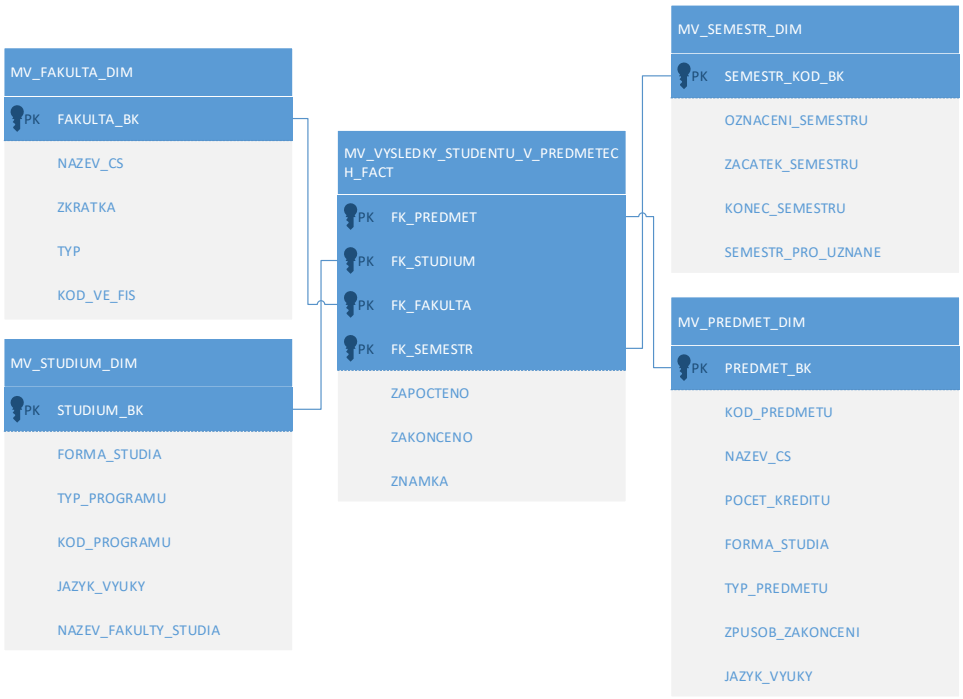
Toto datové tržiště spojuje tři dimenze a jednu faktovou tabulku, která obsahuje pouze cizí klíče do dimenzí. V tomto případě není žádná metrika potřebná, protože nás zajímá pouze počet studií. Počet studií získáme agregací nad cizím klíčem FK_STUDIUM nacházejícím se ve faktové tabulce. Schéma tohoto datového tržiště je znázorněno na obrázku 5.1.



Obrázek 5.1: Model star schématu datamartu s počty studií v semestrech

5.2.2 Datové tržiště s výsledky studentů v předmětech

Toto datové tržiště spojuje čtyři dimenze a jednu faktovou tabulku. Faktová tabulka kromě cizích klíčů do dimenzí obsahuje i další atributy (příznaky započteno, zakončeno a získanou známku), na kterých můžeme provádět různé analytické dotazy. Schéma tohoto datového tržiště je znázorněno na obrázku 5.2.



Obrázek 5.2: Model star schématu datamartu s výsledky studentů

5.3 Vizualizace dat z datových tržišť

Pro přístup uživatelů k datům používáme tři různé způsoby. Ke každému způsobu popisuji jaké má výhody a nevýhody a způsob jakým jsem ho implementoval v datovém skladu ČVUT. Tato sekce se dá označit i jako popis využívání Information Delivery Layer (viz 1.3.5) datového skladu ČVUT.

5.3.1 Pentaho BI Server

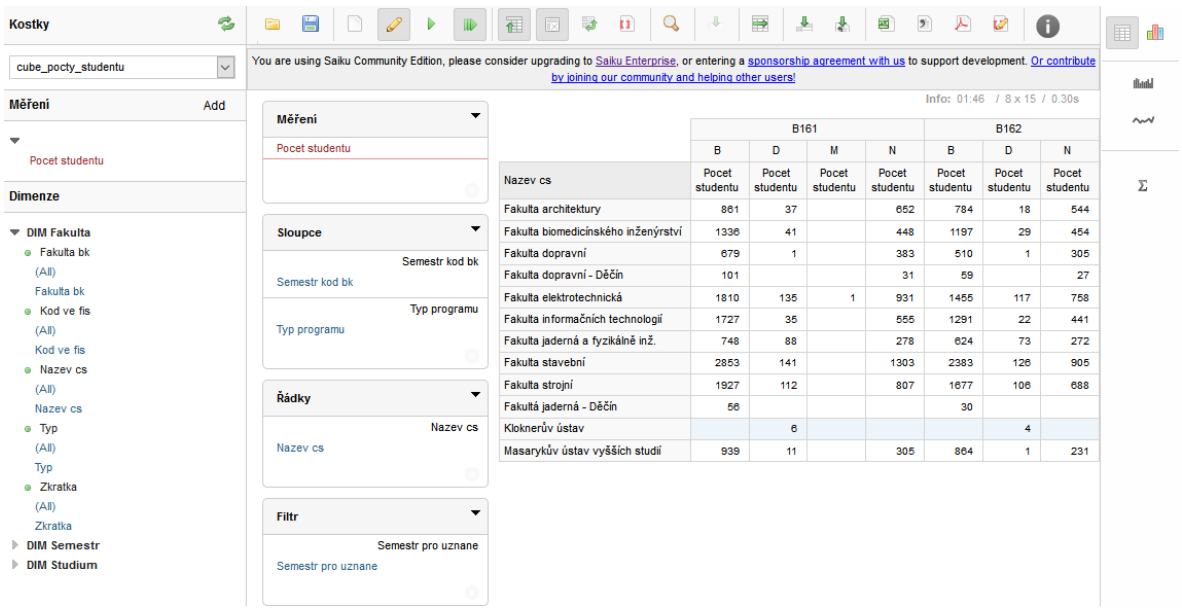
V nástroji Pentaho BI Server je možné provádět analýzy pomocí technologie ROLAP (viz sekce 1.8.2). Tato technologie je implementována pomocí open source platformy Mondrian.

V nástroji jsem vytvořil datové kostky pro implementovaná datová tržiště a nainstaloval analytický modul Saiku. V tomto analytickém modulu je možné provádět analýzy nad datovými kostkami pomocí technologie Mondrian.

Na Pentaho BI Serveru jsem definoval datovou kostku cube_pocty_studentu. Tato kostka se definuje pomocí určení databázových spojení nad databázovým schématem s datovým tržištěm s počty studií v semestrech. Poté jsem nastavil metriku pomocí rozhraní v nástroji Pentaho BI Server. V tomto případě jsem použil počet unikátních hodnot v atributu fk_studium ve faktové tabulce.

Na takto definované kostce bylo již možno provádět analýzy pomocí analytického modulu Saiku. Výstupem tohoto modulu může být report znázorněný na obrázku 5.3. Na tomto obrázku je také vidět uživatelské rozhraní analytického modulu, na kterém je vytvořen dotaz na počet studentů bakalářského, magisterského a doktorského studia v semestrech B161 a B162 na jednotlivých fakultách ČVUT.

Hlavní výhodou tohoto způsobu vizualizace je, že uživatel má možnost vytvářet si jednotlivé reporty sám v nástroji dle jeho požadavků (Pentaho BI Server může být dostupný uživatelům pomocí webového rozhraní). Mezi nevýhody patří nutnost věnovat čas vytvoření reportu, základní znalost domény (aby se uživatel dokázal orientovat v jednotlivých obchodních entitách) a limit na počet uživatelů. Tento limit je dán volností, se kterou uživatel může vytvářet analytické dotazy (může docházet ke složitým výpočtům, které by server při větším počtu uživatelů nemusel být schopen zvládat). Tento limit se může lišit složitostí implementovaných datových tržišť a hardwarovou specifikací serveru.

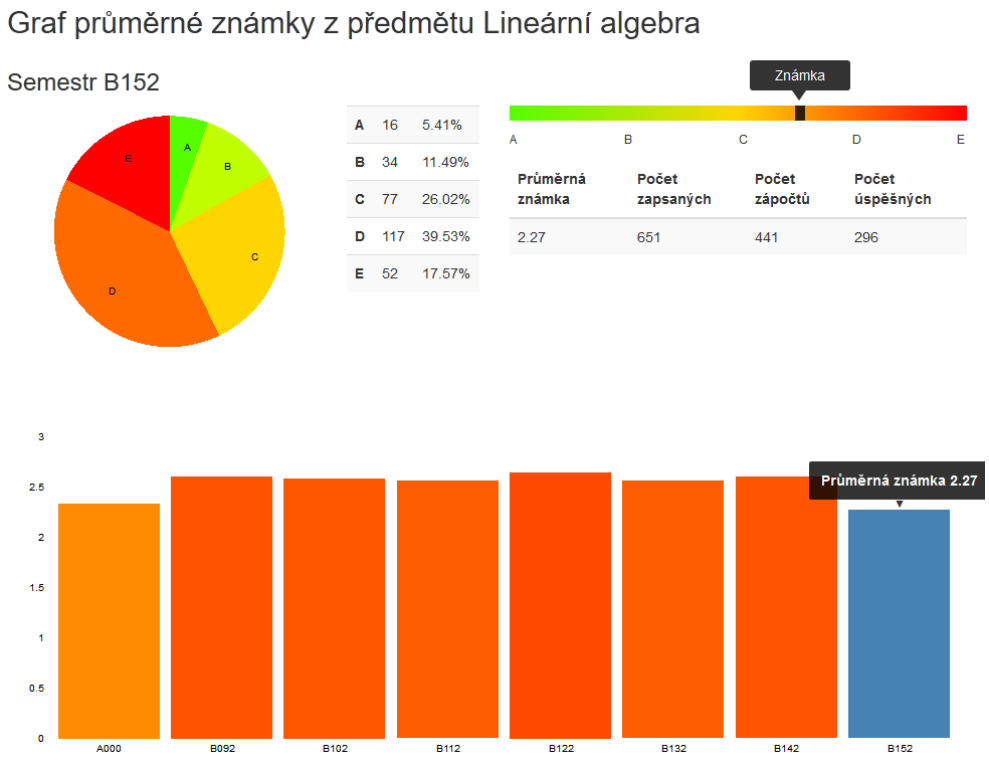


Obrázek 5.3: Analýza pomocí Saiku Analytics v nástroji Pentaho BI Server

5.3.2 EBIE

Vytvořená datová tržiště se využívají i v EBIE[9]. Michal Kopecký vyvinul ve své bakalářské práci [29] interaktivní report, který zobrazuje výsledky předmětů (viz obrázek 5.4). Tento report získává data přímo z datového tržiště s výsledky studentů v předmětech, kde jsou data po každém úspěšném nahrání datového skladu aktualizována. Data obsažené v tomto reportu se dynamicky mění dle datových změn ve zdrojových systémech (v tomto případě v KOSu), z tohoto důvodu jsou vždy aktuální.

Výhodou takto předdefinovaných interaktivních reportů je rychlost a nenáročnost s jakou uživatel potřebné informace získá – jsou tedy vhodné pro přístup velkého počtu uživatelů. Na druhou stranu k vytvoření tohoto reportu je potřeba velké úsilí ze strany vývojářů, protože je nutné ručně implementovat jednotlivé reporty nebo využít placené nástroje jejichž výstupy je nutné někde publikovat. Tyto reporty jsou vytvářeny pro nejčastější požadavky uživatelů – nezobrazují konkrétní požadavky jednotlivců.



Obrázek 5.4: Interaktivní report v EBIE zobrazující průměrné známky v semestrech

[9]EBIE (Extended Business Intelligence Encyclopedia) je portál, který obsahuje předpřipravené reporty a dashboardy s popisem použitých obchodních entit.

5.3.3 Datové sestavy

Nad jednotlivými datovými tržišti je možné provádět analýzy a vytvářet datové sestavy pomocí jazyka SQL přímo nad databází dwh3_access. Výhodou tohoto přístupu je možnost vytvářet datové sestavy podle specifikací konkrétních uživatelů. Pro vytváření těchto datových sestav je nutná dobrá znalost databáze (zde pomáhá dobře definovaná jmenná konvence) a znalost jazyka SQL včetně jeho rozšíření pro analytické dotazy.

Vytvořil jsem ukázkový SQL dotaz 5.3, který vytváří datovou sestavu obdobnou jako je výsledek analýzy pomocí modulu Saiku (viz sekce 5.3.1). Část dat získaných v rámci tohoto SQL dotazu je znázorněna v tabulce 5.1 (výsledek dotazu byl záměrně z důvodu úspory místa zkrácen).

```
1  SELECT
2    fakt.fk_semestr,
3    dim_fakulta.nazev_cs,
4    dim_studium.typ_programu,
5    count(DISTINCT fakt.fk_studium) AS pocet_studentu
6  FROM dm_pocet_studii_v_semestrech.mv_vysledky_studentu_v_predmetech_fact fakt
7    LEFT JOIN dm_pocet_studii_v_semestrech_live.mv_fakulta_dim dim_fakulta
8      ON fakt.fk_fakulta = dim_fakulta.fakulta_bk
9    LEFT JOIN dm_pocet_studii_v_semestrech_live.mv_studium_dim dim_studium
10     ON fakt.fk_studium = dim_studium.studium_bk
11 WHERE fakt.fk_semestr IN ('B161', 'B162')
12 GROUP BY fakt.fk_semestr, dim_fakulta.nazev_cs, dim_studium.typ_programu;
```

SQL 5.3: Analytický dotaz pro získání počtu studentů v semestrech B161 a B162

fk_semestr	nazev_cs	typ_programu	pocet_studentu
B161	Fakulta informačních technologií	B	1727
B161	Fakulta informačních technologií	D	35
B161	Fakulta informačních technologií	N	555
B162	Fakulta informačních technologií	B	1291
B162	Fakulta informačních technologií	D	22
B162	Fakulta informačních technologií	N	441

Tabulka 5.1: Výstup analytického dotazu omezený na FIT

Závěr

Zadáním diplomové práce bylo navrhnout a implementovat datové vrstvy datového skladu ČVUT. Tento úkol se skládal z několika částí.

První částí bylo definování databázové jmenné konvence určené pro databázové objekty používané v datovém skladu ČVUT.

Druhou částí bylo navržení datového modelu centrální databáze datového skladu, který integroval data ze zdrojových systémů KOS, Anketa ČVUT a Závěrečné práce.

Třetí částí byl návrh architektury datového skladu a nalezení způsobu implementace datových vrstev. V této části bylo také potřeba navrhnout propojení dvou databází, které jsou součástí architektury datového skladu.

Funkcionalitu implementovaného řešení jsem otestoval pomocí vizualizace dat z datového tržiště v nástroji Pentaho BI Server a poskytováním datových zdrojů pro interaktivní reporty umístěné v portálu EBIE. Jelikož jsou jednotlivé datové vrstvy na sobě závislé a datová tržiště jsou závislá na všech předchozích datových vrstvách, tak vizualizací dat otestujeme celé implementované řešení.

Zadání diplomové práce jsem splnil. Největším přínosem této práce je navržení databázové jmenné konvence a vytvoření datového modelu centrální databáze, který je možné rozšiřovat o další zdrojové systémy. Praktická část této práce je již nasazena v produkčním prostředí datového skladu ČVUT. Tento datový sklad již poskytuje data, která jsou využívána pro analytické zpracování.

Literatura

[1] Kotlář, R.: *Datový sklad ČVUT - způsoby datové integrace*. Diplomová práce, České vysoké učení technické v Praze, Fakulta informačních technologií, Praha, 2017.

[2] Gála, L.; Pour, J.; Šedivá, Z.: *Podniková informatika*. Praha: Grada Publishing, třetí vydání, 2015, ISBN 9788024754574.

[3] Horakova, M.; Skalska, H.; Olszak, C. M.; aj.: Business Intelligence and Implementation in a Small Enterprise. *Journal of Systems Integration (...*, ročník 4, č. 2, 2003: s. 50–62, ISSN 1804-2724, doi:18042724.

[4] Inmon, W. H.; Strauss, D.; Neushloss, G.: *DW 2.0*. Boston: Morgan Kaufmann, 2008, ISBN 9780123743190.

[5] Ranjan, J.: Business Intelligence: Concepts, Components, Techniques and Benefits. *Journal of Theoretical and Applied Information Technology*, ročník 9, 2009: str. 60, ISSN 18173195, doi:10.2139/ssrn.2150581. Dostupné z: `http://jatit.org/volumes/research-papers/Vol9No1/9Vol9No1.pdf`

[6] Sirůček, P.: *Hospodářské dějiny a ekonomické teorie*. Slaný: Melandrium, vyd. 1. vydání, 2007, ISBN 9788086175034.

[7] Devens, R.: *Cyclopædia of Commercial and Business Anecdotes: Comprising Interesting Reminiscences and Facts, Remarkable Traits and Humors ... of Merchants, Traders, Bankers ... Etc. in All Ages and Countries*. číslo sv. 1 in Cyclopædia of Commercial and Business Anecdotes, D. Appleton, 1865. Dostupné z: `https://books.google.cz/books?id=9MspAAAAYAAJ`

[8] Luhn, H.: A Business Intelligence System. *IBM Journal of Research and Development*, ročník 2, č. 4, 1958: s. 314–319, ISSN 0018-8646, doi:10.1147/rd.24.0314. Dostupné z: `http://altaplana.com/ibmrd0204H.pdf`

[9] Goewey, B.: The History & Impact of Business Intelligence in the 21st Century [online]. 2015, [cit. 2017-02-28]. Dostupné z: http://www.datamensional.com/a-history-and-impact-of-business-intelligence-in-the-21st-century/

[10] Elena, C.: Business intelligence. *Journal of Knowledge Management, Economics and Information Technology*, ročník 1, 2011: str. 101, ISSN 1050-9135, z0037.

[11] Coronel, C.; Morris, S.; Rob, P.: *Database systems*. Boston, MA: Course Technology/Cengage Learning, 10 vydání, c2013, ISBN 9781111969608.

[12] Hayler, A.: Master data management. , č. September, 2013.

[13] Inmon, W. H.: *Building the data warehouse*. Indianapolis, Ind.: Wiley, čtvrté vydání, c2005, ISBN 9780764599446.

[14] Kimball, R.: *The data warehouse lifecycle toolkit*. New York: Wiley, c1998, ISBN 9780471255475.

[15] Ariyachandra, T.; Watson, H.: Key organizational factors in data warehouse architecture selection. *Decision Support Systems*, ročník 49, č. 2, 2010: s. 200–212, ISSN 01679236, doi:10.1016/j.dss.2010.02.006. Dostupné z: http://dx.doi.org/10.1016/j.dss.2010.02.006

[16] Williams, P.: A Short History of Data Warehousing [online]. 2012, [cit. 2017-02-25]. Dostupné z: http://www.dataversity.net/a-short-history-of-data-warehousing/

[17] Arnošt, D.: Kontext BI v rámci komerční firmy, Framework architektury BI a BI architektura. přednášky MI-EDW.16, ČVUT FIT, 2017.

[18] Ariyachandra, T.; Watson, H. J.: Which Data Warehouse Architecture Is Most Successful? *Business Intelligence Journal*, ročník 11, č. 1, 2006: s. 4–6, ISSN 00010782, doi:10.1145/1400181.1400213. Dostupné z: http://www.mendeley.com/research/which-data-warehouse-architecture-is-most-successful/

[19] Kimball, R.; Ross, M.: *The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling*. Indianapolis, IN: Wiley, třetí vydání, 2013, ISBN 978-1-118-53080-1.

[20] Alsqour, M.; Matouk, K.; Owoc, M.: A survey of data warehouse architectures - preliminary results. *Proceedings of the Federated Conference on Computer Science and Information Systems*, 2012: s. 1121–1126.

[21] Kimball, R.; Caserta, J.: *The data warehouse ETL toolkit*. Indianapolis, IN: Wiley, první vydání, c2004, ISBN 0764567578.

[22] learndmdwbi: Designing star schema [online]. 2015, [cit. 2017-03-04]. Dostupné z: https://learndatamodeling.com/blog/designing-star-schema/

[23] learndmdwbi: Designing snowflake schema [online]. 2015, [cit. 2017-03-04]. Dostupné z: https://learndatamodeling.com/blog/designing-snowflake-schema/

[24] Martino, A.: Trivadis White Paper Comparison of Data Modeling Methods for a Core Data Warehouse. *Trivadis White Paper*, , č. June, 2014: str. 21.

[25] ISO: Information technology - Metadata registries (MDR) - Part 5: Naming principles. ISO/IEC 11179-5:2015, International Organization for Standardization, Geneva, Switzerland, 2015.

[26] Choudhary, P. K.: Naming Conventions in SQL [online]. 2015, [cit. 2017-04-27]. Dostupné z: http://www.c-sharpcorner.com/UploadFile/f0b2ed/what-is-naming-convention/

[27] Keller, J. J.: An Unemotional Logical Look at SQL Server Naming Conventions [online]. 2015, [cit. 2017-04-27]. Dostupné z: http://www.vertabelo.com/blog/technical-articles/an-unemotional-logical-look-at-sql-server-naming-conventions

[28] Hanada, S.: postgres_fdw [online]. 2016, [cit. 2017-04-27]. Dostupné z: https://www.postgresql.org/docs/9.5/static/postgres-fdw.html

[29] Kopecký, M.: *Návrh a implementace klientské části systému EBIE*. Bakalářská práce, České vysoké učení technické v Praze, Fakulta informačních technologií, Praha, 2016.

Příloha **A**

Seznam použitých zkratek

ACID Atomic Consistent Isolated Durable

BI Business Intelligence

CRM Customer Relationship Management

CSV Comma-separated Values

DCS Decision Support System

DDL Data Definition Language

DMS Document Management System

DSS Decision Support System

EDW Enterprise Data Warehosue

ELT Extract Load Transform

ERP Enterprise Resource Planning

ETL Extract Transform Load

IDL Integrated Data Layer

MOLAP Multidimensional OLAP

OLAP Online Analytical Processing

OLTP Online Transaction Processing

ROLAP Relational OLAP

SCD Slowly Changing Dimension

SQL Structured English Query Language

Příloha B

Jmenná konvence datového skladu

DWH3 jmenná konvence databázového řešení

Business Intelligence Group

České vysoké učení v Praze
Fakulta informačních technologií

Datum vytvoření: 2.6.2016

Datum poslední změny: 25.4.2017
Revize: 1.31

Tato metodika je navržena pro verzi datového skladu DWH3. Struktura celé databáze je vyvíjena pomocí této metodiky a i další verze datového skladu budou z této metodiky vycházet.

Všechny **atributy** a **názvy tabulek** a jiné **databázové objekty** budou psány **velkými tiskacími písmeny v češtině (bez diakritiky)** dle **pravidel** specifikovaných níže, **výjimku** tvoři pouze **databázové procedury a funkce**, které jsou z důvodu přehlednosti p**sané a pojmenovávané v angličtině**. Struktura databáze a druhy jednotlivých schémat jsou pevně dané a je nutné tento návrh při implementaci dodržovat.

Poznámka!!!

Pro současnou verzi skladu budou používána malá písmena (omezení technologie PostgreSQL v kombinaci s dalšími open-source nástroji) pro pojmenovávání databázových objektů. Vše kromě pojmenovávání databázových objektů bude psáno klasicky velkými písmeny. Velká písmena jsou ponechány v projektu pro případné zjednodušení migrace na databázový stroj ORACLE.

Stupně databázového modelu použitého při vývoji datového skladu pomocí konceptuálního nástroje Enterprise Architect.

- **businessový model** - obsahuje businessové vazby; v návrhu chybí technické atributy specifické pro datový sklad, není zde zahrnuta historizace a obsahuje:
    - rozdělení tabulek do businessových entit
    - asociace mezi tabulkami
    - businessový primární klíč
    - indexy nad atributy

- **fyzický model -** jedná se o fyzický model hlavní databáze datového skladu jsou přidány technické atributy a odstraněny veškeré asociace a businessové primární klíče - těm je přidán constraint not null. Tento model je přímo napojen na databázi (změny v modelu se ihned promítnou i do databáze).
    - technické atributy
        - NAZEV_TABULKY_TK (BIGINT) - technický klíč záznamu (sekvence)
        - VERSION (BIGINT) - verze záznamu
        - DATE_FROM (TIMESTAMP)- platnost záznamu od
        - DATE_TO (TIMESTAMP) - platnost záznamu do
        - LAST_UPDATE (TIMESTAMP) - datum poslední změny

Celý projekt bude obsahovat následující databáze s následujícími schématy, které budou pojmenovány dle následující konvence.

DWH3_STAGE

Typy schémat:
PRE_STAGE - datový otisk zdrojového systému pro nahrání do datového skladu, strukturu názvu schématu je následující:

PS_[zkratka/název zdrojového systému]

Pokud název tabulky obsahuje suffix **_ORIG** jedná se o originální verzi tabulky s daty obsaženými ve zdrojovém systému. Takto označená tabulka má i svoje “dvojče” ve schématu PRE_STAGE_CLEAN, kde má tato tabulka opravena data, která jsou technicky nutná k úspěšnému nahrání ETL.

PRE_STAGE_CLEAN - schéma obsahuje technicky vyčištěné tabulky pro nahrání do datového skladu, strukturu názvu schématu je následující:

PSC_[zkratka/název zdrojového systému]

PSC_UDE - speciální typ pre_stage_clean, který obsahuje **U**měle **D**efinované **E**ntity. Toto schéma může používat libovolné tabulky ze schémat PS a PSC. Tabulky se pojmenovájí dle klasické notace, ale nejsou zařazeny do žádné domény. Název se sestavuje podle obsahu dané tabulky (např. T_ZAPSANE_PARALELKY_STUDENTA).

STAGE_INCREMENT - poslední otisk ze zdrojového systému, který byl do skladu nahrán (obsahuje tabulky z PRE_STAGE i PRE_STAGE_CLEAN), strukturu názvu schématu je následující:

SI_[zkratka/název zdrojového systému]

SI_UDE - speciální typ stage_incrementu obsahuj poslední otisk ze schématu PSC_UDE

DWH3_TARGET
schémata:
CISELNIKY - číselníky s hodnotami obsaženými v hlavní databázi
PUBLIC - hlavní databáze datové skladu
SEMANTIC - sémantická vrstva (definice businessových pravidel nad datovými entitami)
METADATA - metada o běhu ETL procesů apod.

DWH3_ACCESS
schémata:
SEMANTIC - propojení s hlavní databází pomocí technologie foreign data wrapper
DM_[Zkratka/název datamartu] - jednotlivé datamarty dle konkrétních požadavků

Tabulky

Název každé tabulky je definován **jednotným číslem velkými tiskacími písmeny**, kde jednotlivá slova jsou odděleny podtržítkem. Šablona pro jednotný název tabulek je:

[prefixy]_[doména]_NAZEV_TABULKY_[suffixy]

Př.: T_OSOB_OSOBA

Prefixy

- T - klasická tabulka
- MV - materializovaný pohled
- V - pohled
- FT - cizí tabulka (foreign table)

Suffixy

- DIM - označení dimenzionální tabulky v případě dimenzionálního schématu
- FACT - označení faktové tabulky v případě dimenzionálního schématu
- REL - označení, že se jedná o relační tabulku (viz. asociace M:N)
- NOHIST - tabulka, která se v datovém skladu nehistorizuje - nemá technické atributy

Zvláštní tabulky:

- prefix:
    - TMP - dočasná tabulka, která slouží pouze pro běh nějaké funkce...
    - WRK - tabulka, která slouží pro běh funkce (neobsahuje logiku)

Prefixy i subfixy se mohou libovolně skládat za sebou (odděleny podtržítky), aby bylo možné správně vyjádřit vlastnosti jednotlivých tabulek.

Poznámka: Všechny názvy tabulek musí být unikátní i bez názvů domén - domény pouze zpřehledňují orientataci v databázovém schématu.

Doména

Je čtyřpísmenná zkratka domény, ve které se daná tabulka nachází. Toto usnadňuje orientaci v databázi (jednotlivé domény jsou uloženy v tabulce metadat). Tato část názvu tabulky se vyplňuje pouze pro schéma “dwh” - centrální databáze datového skladu.

Atributy

Název atributů je definován **jednotným číslem a velkými tiskacími písmeny**, kde jednotlivá slova jsou oddělena podtržítkem. Šablona pro jednotný název atributů je:

[prefix]_NAZEV_ATRIBUTU_[suffix]

prefixy

- FK - v případě, že se jedná o cizí klíč

suffix

- BK - businessový klíč používaný ve zdrojovém systému (business PK tabulky)
- TK - technický klíč vygenerovaný v rámci datového skladu sekvencí (PK tabulky)
- ID - jednoznačný identifikátor, který identifikuje řádek v tabulce (nejedná se o business klíč používaný v rámci businessové domény)

Nad všemi atributy se suffixem BK, TK nebo ID je drženo integritní omezení NOT NULL.

Businessový klíč (BK) je definován dle následující šablony (část specifikace klíče nám umožňuje definovat businessový klíč pomocí ntice nebo nám poskytuje možnost blíže specifikovat klíč v rámci businessové domény - viz businessový klíč tabulky OSOBA - peridno)

[prefix]_NAZEV_TABULKY_[specifikace klíče]_BK

Př.: OSOBA_PERIDNO_BK

Technický klíč (TK) je definován dle následující šablony a jedná se vždy o BIGINT, který je vytvářen dle sekvence:

NAZEV_TABULKY_TK

Př.: OSOBA_TK

Asociace

Struktura cizího klíče:

V architektuře datového skladu, který používáme, jsou všechny vztahy mezi tabulkami řešeny pouze pomocí businessových klíčů (toto je důležité z hlediska historizace používané v našem návrhu - hlavní databáze v 3nf s CDC).

FK_TABULKAODKUD_[specifikace klíče]

Pozn.: Pokud je cizí klíč zároveň businessovým klíčem dané tabulky, je za něj připojen suffix _BK.

Př.: FK_OSOBA_PERIDNO_BK (tabulka T_UCIT_UCITEL)

Vztah M:N:

Tento vztah se řeší přidáním nové vazební tabulky, kterou pojmenujeme dle následující specifikace:

[prefix]_TABULKAODKUD_TABULKAKAM_REL

Prefix je standardně definován v sekci Tabulky a jedná se pouze o určení typu databázového objektu. Atributy tabulky se skládají ze dvou cizích klíčů, které jsou současně i BK. Jmenná konvence těchto klíčů je definovaná v sekci Struktura cizího klíče.

Př.: tabulka: T_UCITEL_BEHPREDMETU_REL,
atributy: FK_UCITEL_PERIDNO_BK, FK_BEHPREDMETU_BK

Ostatní vztahy:

Ostatní vztahy mezi tabulkami nejsou již nijak řešeny pomocí zvláštní jmenné konvence. Používá se pouze konvence používaná pro cizí klíč, která je definovaná výše.

Další databázové objekty

Sekvence

Sekvence nazýváme dle následující specifikace, kde popis je volitelná položka, která slouží k lepšímu odlišení v případě, že je nutné mít více sekvencí nad jednou tabulkou (např. _TK - sekvence se vztahuje k TK tabulky):

SEQ_NAZEVTABULKY_[popis]

Př.: SEQ_OSOBA_TK

Constrainty

Constrainty jsou definovány následující specifikací:

CON_NAZEVTABULKY_NÁZEV_SLOUPCE

Př.: CON_OSOBA_PERIDNO

Triggery

Constrainty jsou definovány následující specifikací, trigger se váže vždy k názvu tabulky a je možné uvést i popis, který slouží pro lepší identifikaci triggeru nebo možnosti mít více trigerrů nad tabulkou:

TR_NAZEVTABULKY_[popis]

Indexy

Nad businessovým klíčem

BK_NAZEV_TABULKY

Př.: BK_OSOBA
poznámka: nepůjde o unikátní indexa to z důvodu použití CDC

Nad technickým klíčem

TK_NAZEV_TABULKY

Př.: TK_OSOBA

V případě unikátního indexu

Unikátní indexy popisujeme standardně dle šablon níže. V případě, že index obsahuje pouze samostatný sloupec využijeme horní notaci, pokud je unikátní index definován jako ntice atributů, používá se dolní notace (popis slouží k odlišení konkrétní ntice nad jednou tabulkou).

UX_NAZEVTABULKY_SLOUPEC

UX_NAZEVTABULKY_[popis]

Př.: UX_OSOBA_USERNAME

V případě standardního indexu

Standardní indexy popisujeme standardně dle šablon níže. V případě, že index obsahuje pouze samostatný sloupec využijeme horní notaci, pokud je standardní index definován jako ntice atributů, používá se dolní notace (popis slouží k odlišení konkrétní ntice nad jednou tabulkou).

IX_NAZEVTABULKY_SLOUPEC

IX_NAZEVTABULKY_[popis]

Př.: IX_OSOBA_IDENTIFIKACE

Funkce

Funkce a procedury se budou psát a pojmenovávat celé v angličtině z důvodu přehlednosti kódu. Komentáře funkcí a procedur se budou psát celé česky. Každá funkce, či procedura musí obsahovat komentář, který bude popisovat dostatečně výstižně, co dělá, jaký je její vstup a výstup.

Funkce se pojmenovávají podle této šablony:

FC_PREFIX_[FUNCTION_NAME, or short describtion]

prefixy

- TMP - dočasně vytvořená procedura nebo funkce, nepoužívá se při žádném procesu použitém v systému DWH3
- HST - funkce, které slouží pro správné načítání dat při tvorbě semantické vrstvy (historizace pro semantickou vrstvu)
- INC - funkce, které slouží ve Stage Incrementu
- DBL - funkce, pro správný chod databázového propojení

Ukládání jednotlivých funkcí a procedur

Vše se bude ukládat do určeného schématu, dle využití. Procedury a funkce ukládáme dle následujícího vzorce:

STAGE

Schéma	**Druh**
PRE_STAGE_CLEAN	Funkce spojené s technickým čištěním.
PUBLIC	Funkce pro stage_increment s prefixem INC - nezávislé na zdrojový systém

TARGET

Schéma	**Druh**
SEMANTIC	Funkce pro vytváření semantické vrstvy a historizační funkce (prefix HST)
PUBLIC	Databázové propojení a servisní funkce.

Konstanty použité v datovém skladu

Defaultní hodnoty pro prázdné atributy
-1 - no data (ze zdrojového systému nelze získat data -> chyba zdrojového systému, kvalita dat)
-2 - not implemented (tento atribut ještě nebyl implementován a tedy jeho hodnota není naplněna v rámci ETL)

Defaultní hodnoty (nastaveny na databázové úrovni)
DATE_TO = 2199-12-31 00:00:00.000000
DATE_FROM = 1900-01-01 00:00:00.000000

Defaultní hodnoty příznaků
positivní (true/ano) = 1
negativní (false/ne) = 0

Použité zkratky:
CDC - changing data capture
ETL - extract transform load
3nf - třetí normální forma

Příloha C

Databázová funkce pro vytvoření fyzického modelu centrální databáze

```
1  CREATE OR REPLACE FUNCTION public.FC_GENERATE_TECHNICAL_COLUMNS
2  (schema_name VARCHAR(100))
3    RETURNS VOID AS $$
4  DECLARE
5      curs_tables_in_schema CURSOR FOR SELECT *
6                                       FROM pg_catalog.pg_tables
7                                       WHERE schemaname = schema_name;
8    table_rec                    RECORD;
9    table_name                   VARCHAR(150);
10   table_name_short             VARCHAR(150);
11   table_name_length            INTEGER;
12   column_rec                   RECORD;
13   column_name                  VARCHAR(150);
14   column_name_length           INTEGER;
15   constraint_rec               RECORD;
16   bks_for_index_creation       TEXT;
17   bks_for_index_creation_count INTEGER;
18
19 BEGIN
20   FOR table_rec IN curs_tables_in_schema
21   LOOP
22
23     table_name := table_rec.tablename;
24     table_name_length := char_length(table_name);
25     bks_for_index_creation_count = 0;
26     bks_for_index_creation = '';
27
28     IF substring(table_name FROM table_name_length - 3) = '_rel'
29     THEN
30       table_name_short = substring(table_name FROM 3 FOR
31       table_name_length - 6);
```

```
32
33      ELSE
34        table_name_short = substring(table_name FROM 8);
35      END IF;
36
37      FOR constraint_rec IN SELECT *
38                            FROM pg_catalog.pg_constraint
39                            WHERE contype IN ('f', 'p') AND
40                                  pg_constraint.conrelid = (schema_name
41                                  || '.' || table_name) :: REGCLASS
42      LOOP
43
44        EXECUTE 'ALTER TABLE ' || schema_name || '.' || table_name || ' DROP
45        CONSTRAINT IF EXISTS ' ||constraint_rec.conname || ' CASCADE;';
46
47      END LOOP;
48
49      FOR column_rec IN SELECT *
50                        FROM pg_catalog.pg_attribute
51                        WHERE
52                          attrelid = (schema_name||'.'||table_name)::REGCLASS
53                          AND attnum > 0 AND NOT attisdropped
54      LOOP
55        column_name = column_rec.attname;
56        column_name_length = char_length(column_name);
57
58        IF substr(column_name, column_name_length - 2) = '_bk'
59        THEN
60          bks_for_index_creation = bks_for_index_creation||column_name||', ';
61          EXECUTE 'CREATE INDEX ix_' || replace(table_name_short, '_', '')
62           || '_' || replace(substring(column_name FOR column_name_length-3)
63            , '_', '') || ' ON ' || schema_name || '.' || table_name
64           || ' (' || column_name || ');';
65          bks_for_index_creation_count = bks_for_index_creation_count + 1;
66
67        ELSEIF substr(column_name, column_name_length - 2) = '_id'
68          THEN
69            EXECUTE 'CREATE INDEX ix_' || replace(table_name_short, '_', '')
70            || '_' || replace(substring(column_name FOR column_name_length-3)
71            , '_', '') || ' ON ' || schema_name || '.' || table_name ||
72            ' (' || column_name || ');';
73        END IF;
74
75      END LOOP;
76
77      EXECUTE 'CREATE SEQUENCE ' || schema_name || '.seq_' ||
78      replace(table_name_short, '_', '') || ' START 1 INCREMENT 1;';
79
80      EXECUTE
81      'ALTER TABLE ' || schema_name || '.' || table_name ||
82      ' ADD COLUMN ' || table_name_short || '_tk BIGINT CONSTRAINT pk_'
83      || table_name_short || ' PRIMARY KEY DEFAULT nextval(''' ||
84      schema_name || '.seq_' || replace(table_name_short, '_', '') || '''),
85      ADD COLUMN VERSION BIGINT,
```

```
86      ADD COLUMN DATE_FROM TIMESTAMP,
87      ADD COLUMN DATE_TO TIMESTAMP,
88      ADD COLUMN LAST_UPDATE TIMESTAMP;';
89
90      IF bks_for_index_creation_count > 1
91      THEN
92        EXECUTE 'CREATE INDEX ix_' || replace(table_name_short, '_', '')
93        || '_bk ON ' || schema_name || '.' ||
94        table_name || ' (' || substring(bks_for_index_creation
95        FROM 0 FOR length(bks_for_index_creation) - 1) ||');';
96      END IF;
97
98      RAISE NOTICE 'CHANGES IN TABLE %.% ARE COMPLETED',
99      schema_name, table_name;
100
101   END LOOP;
102
103   RAISE NOTICE 'ALL TABLES IN SCHEMA % HAVE BEEN CHANGED',
104   schema_name;
105
106 END;
107 $$ LANGUAGE 'plpgsql';
108
109 ALTER FUNCTION public.FC_GENERATE_TECHNICAL_COLUMNS( VARCHAR )
110 OWNER TO big;
```

Příloha D

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `src`
  - `impl` ..... zdrojové kódy implementace
    - `jmenna_konvence.pdf` jmenná konvence použitá v datovém skladu
    - `dwh3_target_model.pdf` .... databázový model centrální databáze datového skladu
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF