Univerzita Hradec Králové
Fakulta informatiky a managementu
Katedra informatiky a kvantitativních metod

Big Data a Business Intelligence

Bakalářská práce

Autor: Jan Bradáč
Studijní obor: Aplikovaná informatika

Vedoucí práce: prof. RNDr. Hana Skalská, CSc.

Hradec Králové květen 2016

Prohlášení:

Prohlašuji, že jsem bakalářskou práci zpracoval samostatně a s použitím uvedené literatury.

V Hradci Králové dne 2. 5. 2016 Jan Bradáč

Poděkování:

Děkuji vedoucí bakalářské práce prof. RNDr. Haně Skalské, CSc. za metodické vedení práce, cenné rady a trpělivost.

Anotace

Hlavním cílem této bakalářské práce je podat ucelený přehled z oblastí big data a business intelligence. Kromě popisu těchto hlavních pojmů je představen systém Hadoop, který je aktuálním trendem v big data řešeních. Popis Hadoop se soustředí především na klíčové komponenty HDFS a MapReduce, ale jsou zde stručněji popsány i další rozšiřující moduly. Práce také obsahuje definice dat a jejich typů, popis zdrojů dat a jejich integraci pro analýzu. Pozornost je také věnována kvalitě dat. Jsou zde naznačeny problémy s nekvalitními zdrojovými daty, možnosti jejich řešení a příprava dat před analýzou. Je zde vysvětlen data mining a předvedena praktická ukázka vytvoření prediktivního modelu v softwaru IBM SPSS Modeler. Jsou zde také probírány vlastnosti big data platformy spolu s řešením pomocí Lambda architektury.

Annotation

Title: Big Data and Business Intelligence

The main goal of this Bachelor thesis is to give a compact overview of Big Data and Business Intelligence. Besides the description of these main concepts there is Hadoop system introduced, which is the actual trend in Big Data solutions. Hadoop description is mainly focused on the key components HDFS and MapReduce but there are also other extension modules described briefly. This thesis also contains definitions of data and data types, description of data sources and their integration for analysis. Attention is also paid to data quality. There are hinted problems with source data of poor quality, their solution possibilities and data preparation before analysis. There is Data Mining explained and practical example of creation of predictive model with IBM SPSS Modeler software demonstrated. There are also characteristics of Big Data platform discussed along with Lambda Architecture solution.

Obsah

Seznam obrázků

Seznam tabulek

1 Úvod

Vše se odvíjí od všudypřítomných dat, která mají svá rozdělení, musí se zachytávat, skladovat, transformovat a třídit. Tyto údaje je možné zkoumat a analyzovat z různých dimenzí, různými nástroji nebo metodami, což přináší nové informace pro učinění rozhodnutí. S rozvojem informačních technologií a jejich snadné dostupnosti vzrůstá množství produkovaných dat, které je těžko představitelné. Vznikají v tomto směru nové možnosti, otázky a problémy. Pro jejich řešení jsou potřebné určité znalosti a systémy. V oblasti informačních technologií jsou často zmiňovány termíny big data (BD), business intelligence (BI) nebo data mining (DM), což jsou oblasti zabývající se problémy rozsáhlých dat. Odborná literatura je často zaměřena jen na jednu z řešených oblastí, navíc může být sepsána z jedné určité perspektivy a i tak se může jednat o velmi rozsáhlé množství informací. Bylo by asi nemožné vytvořit něco ve smyslu „Lexikonu všech kouzel“, kde by byly obsaženy veškeré postupy, procedury, technologie, vlastnosti, struktury a problematika, uvedené za všech možných situací a různých perspektiv a vše vměstnat do rozsahu bakalářské práce. Proto je tato práce spíše jakýmsi povrchovým průřezem napříč zvolenými tématy a celkově je popisuje, vysvětluje jejich význam a souvislosti, naznačuje jejich komplexitu, uvádí příklady a může sloužit jako úvod do hlubšího studia vybraných oblastí.

2 Cíl práce

Hlavním cílem je seznámení s aktuálními trendy ve světě informačních technologií, mezi které patří oblasti big data, business intelligence a data mining. V přehledu sestaveném z vybrané literatury představit společně s těmito oblastmi také některé technologie, které se v nich využívají, zejména systém Apache Hadoop, který je v současnosti populární a využívaný v praktických řešeních. Dalším cílem je charakterizovat data, vysvětlit jejich význam a naznačit jejich možnosti využití, také vysvětlit problémy při zpracování dat a nastínit jejich možná řešení. Pro názornost a hlubší vhled do zvolených témat je dalším cílem předvést ukázku datového modelování v IBM SPSS Modeler a také uvést architekturu, vhodnou pro big data, do které zapadá i Apache Hadoop.

3 Metodika zpracování

Metodika zpracování vychází ze zvolené odborné literatury, která souvisí s řešenými tématy a navzájem se doplňuje. Použitá literatura je v českém i anglickém jazyce, jak v tištěné, tak v elektronické formě. Podle vlastního uvážení jsou zachovány některé anglické termíny s uvedením českého překladu nebo popisu. Vysvětlující obrázky se nacházejí mezi textem, ke kterému se vztahují, pro lepší pochopení a orientaci. Z použitých literárních zdrojů je sestaven přehled, který vysvětluje a popisuje stěžejní a související témata s cílem naznačení jejich problematiky, rozsahu a komplexnosti a vytvoření celkového nadhledu. Zpočátku je práce zaměřena na data, která jsou pojítkem všech obsažených oblastí a technologií. Drží se spíše v obecnější rovině a tradičních systémů s naznačením některých příkladů. V 7. kapitole se práce začíná více orientovat na popisy konkrétních technologií a příkladů a jejich principů.

4 Data a typy dat

Data reprezentují reálné objekty, respektive hodnoty jejich atributů, které můžeme nějakým způsobem zpracovat, pokud máme potřebné informace nebo znalosti. Data mohou mít například formu obrázku, zvuku, symbolu nebo textu (nemusí mít digitální charakter). Bez relevantních informací nebo znalostí nemusí mít samotná data pro člověka (nebo stroj) žádnou hodnotu. Například text napsaný v cizím jazyce jsou nějaká data, ale informaci přinesou jen člověku (stroji), který bude tomuto jazyku rozumět. Jednoznačné definice pro informace a znalosti není jednoduché určit, ale, zjednodušeně řečeno, informace dávají datům nějaký smysl a pomocí znalostí může být na základě dat a informací učiněno nějaké rozhodnutí. Data se dají rozlišit podle jejich struktury: strukturovaná, nestrukturovaná a semistrukturovaná.

(Čech a Bureš, 2009)

- **Strukturovaná** data jsou obvykle uložena v relačních databázích (RDBMS - Relational Database Management Systems). Každý uložený záznam dodržuje předem vymezenou strukturu. Může se jednat o data z transakčních systémů, RFID čteček, různých snímačů a aplikací pro zpracování dat. Strukturovaná data neukazují širší souvislosti a většinou nemají tak velký informační potenciál jako data nestrukturovaná.

  (Čech a Bureš, 2009)
- **Nestrukturovaná** data jsou zasazena do širšího informačního kontextu, a díky tomu je možné z jejich analýzy objevit důležité poznatky. Nejedná se však o triviální záležitost. Za nestrukturovaná data lze označit různé dokumenty, emailové zprávy, reporty nebo články. Kromě textové formy dat se jedná také o obrázky, multimediální sobory a další. Tento druh dat tvoří největší část ukládaných dat.

  (Čech a Bureš, 2009)
- **Semistrukturovaná** data nezapadají do strukturovaného databázového modelu, ale nějakým způsobem (částečně) strukturovaná jsou. Například sobory XML, CSV nebo JSON.

  (Pierson a Porway, 2015)

4.1 Metadata

Metadata jsou označována jako data o datech (o zdrojích informací). Vyskytují se jak v reálném světě, tak v elektronické podobě. Mohou popisovat jediný zdroj, kolekci zdrojů nebo část celku (například obrázek v článku). Umístěna mohou být přímo ve zdroji nebo i mimo něj. V reálném světě je například kniha zdroj informací a obsahuje metadata, jako jméno autora, datum vydání a počet stran. V elektronické podobě obsahuje například soubor uložený v počítači metadata, jako datum vytvoření souboru, jeho velikost, umístění… HTML soubory mohou obsahovat v hlavičce klíčová slova, popis obsahu, autora… Sloupec databázové tabulky má určený název, datový typ, velikost... To všechno jsou metadata. Mohou být potřebná jak pro člověka, tak pro stroj a těmito entitami jsou i tvořena.

(Understanding metadata, 2004)

Zdroj Understanding metadata (2004) rozděluje metadata do tří skupin: deskriptivní, strukturální a administrativní.

- **Deskriptivní** metadata popisují zdroj za účelem snadné identifikace a vyhledání. Například v případě knihy: název, autor, klíčová slova, abstrakt.
- **Strukturální** metadata udávají interní strukturu objektu (stránky, kapitoly, obsah...)
- **Administrativní** metadata jsou technického charakteru a pomáhají spravovat zdroj. Například informace o tom, kdy byl zdroj vytvořen, typ souboru, přístupová práva a podobně)

(Understanding metadata, 2004)

Na první pohled by se mohlo zdát, že metadata jsou „méně než data“, že nemají takovou důležitost nebo přínos. Ve skutečnosti však významně ovlivňují a usnadňují práci s „hlavními“ daty. Díky nim je možné údaje mnohem efektivněji vyhledávat, třídit, ukládat, zjistit jejich původ nebo čas vytvoření. Poskytují důležité informace, bez kterých by mohla být data neužitečná či bezcenná. Důležitost metadat dokazuje i zdroj Kimball a Caserta (2004), kde je pro metadata

vyčleněna samostatná kapitola a jsou popsána velmi podrobně z pohledu skladování dat, což je také součást tématu této práce.

4.2 Big data

Pojem big data není jen „módní pojem“ (anglicky buzzword) při dnešních trendech, ale rychle se vyvíjející koncept v informačních technologiích a managementu dat. Zároveň je ale těžké tento pojem přesně definovat, protože co IT expert, to jiný popis (Reynolds, 2016). Podle Piersona a Porwaye (2015) se jedná o data, která překračují výpočetní kapacitu konvenčních databázových systémů, přibývají příliš rychle nebo nemají vhodnou strukturu pro tradiční databázové architektury. Manoochehri (2014) uvádí, že podle George Dysona (historika vědy) se jedná o big data, když cena za výpočetní prostředky potřebné ke zpracování údajů je nižší než hodnota, kterou nám zpracování dat přinese. Poradenská firma Gartner definuje big data jako pojem označující velká množství dat, která již není možno běžnými softwarovými nástroji zachytávat, zpracovávat nebo spravovat v rozumném čase (Dolák, 2011). Dále Dolák (2011) uvádí, že tedy nezáleží jen na množství (objemu) údajů, ale také na rychlosti jejich množení, přenášení a různorodosti. Marr (2015) píše, že termínu big data je těžké porozumět, protože, v závislosti na úhlu pohledu, může znamenat pro různé lidi různé věci. Záleží tedy, jestli se k big data přistupuje například z pohledu technologického, obchodního nebo průmyslového. V češtině je termín big data někdy překládán jako „rozsáhlá data“ (Dolejš, 2012) nebo „veledata“ (Mayer-Schönberger a Cukier, 2014).

Rozsáhlými daty se zabývá například společnost Google, která každý den zpracovává více než 24 petabajtů dat, nebo Facebook, na který jeho uživatelé nahrají každou hodinu přes 10 milionů fotografií (Mayer-Schönberger a Cukier, 2014). Dále například Národní úřad pro oceán a atmosféru (NOAA), Národní úřad pro letectví a kosmonautiku (NASA), různé energetické, telekomunikační, farmaceutické společnosti a další (Dolák 2011). Díky těmto nasbíraným údajům a jejich zpracování, v podobě různých druhů analýz, získávají organizace cenné informace, které jim umožňují vylepšit své služby, předpovídat budoucí vývoj nebo usnadnit rozhodovací procesy. Například, když se v roce 2009 objevil nový virus

chřipky, Google ve spolupráci s agenturou CDC (Centers for Disease Control and Prevention), za pomoci dostupných dat z minulosti, vyvinul způsob, jak na základě lidmi zadávaných vyhledávacích dotazů předpovědět oblasti šíření viru v téměř reálném čase (Mayer-Schönberger a Cukier 2014).

Big data se také charakterizují podle anglických výrazů začínajících písmenem V, nicméně použité informační zdroje se rozcházejí v jejich počtu. Anderson (2015) předkládá tři V, tedy volume (objem), velocity (rychlost) a variety (různorodost). Tato tři V se shodují s ostatními zdroji. Pierson a Porway (2015) i Hurwitzová et al. (2013) popisují čtyři V, ale Hurwitzová et al. jako čtvrté V udávají veracity (věrohodnost) a Pierson a Porway oproti tomu value (hodnota). Marr (2015) popisuje všech pět zmíněných výrazů jako pět V a uvádí value (hodnotu) jako nejdůležitější z nich. Rijmenam (2013) a McNultyová (2014, čerpá také od Rijmenama) informují o sedmi V, kde, k výše uvedeným pěti, přidávají navíc variability (proměnlivost, nestálost) a visualization (vizualizace).

Je vidět, že big data jsou stále vyvíjejícím se trendem a jejich definice se mohou lišit v závislosti na jednotlivých expertech nebo úhlech jejich pohledu. Press (2013) uvádí, že termín „big data“ se v digitální knihovně ACM (http://dl.acm.org) poprvé objevil v odborné práci z října roku 1997 (Cox a Ellsworth, 1997), která se zabývá problémy vizualizace dat. Je v ní zmíněno, že data dnes (v roce 1997), při vizualizaci CFD (Computational Fluid Dynamics - vizualizace chování tekutin a plynů), mohou překročit i 100 GB. Množství dat, která už není schopna pojmout paměť počítače (jak operační, tak dlouhodobá) je zde označeno za „problém velkých dat“. Definice big data se tedy mění i podle vývoje informačních technologií, jejich možností a potřeby řešení nově vzniklých problémů. Následuje popis výše zmíněných sedmi V:

- **Volume** (objem) se vztahuje k rozsáhlému množství dat, která jsou vytvářena. Jejich množství se neustále zvyšuje, což dokazuje i fakt, že stejné množství dat co se vygenerovalo do roku 2000, vytváříme dnes každou minutu (Marr, 2015). V případě neočekávaného nebo nepředvídatelného množství dat je třeba zajistit vhodnou infrastrukturu, která je založena na distribuovaném výpočetním modelu. Data mohou být fyzicky uložena na

mnoha místech a propojena počítačovou sítí. Využívá se distribuovaný souborový systém, různé analytické nástroje a aplikace pro rozsáhlá data.

(Hurwitzová et al., 2013)

- **Velocity** (rychlost) označuje rychlost, kterou se data generují a pohybují. Technologie dnes umožňují provádět analýzu dat už při jejich vytváření bez toho, aby se kdy uložila do databází. Například obchodní systémy během milisekund analyzují údaje ze sociálních sítí, na jejichž základě se může rozhodovat o prodeji nebo nákupu akcií.

(Marr, 2015)

- **Variety** (různorodost) odkazuje k různým druhům dat. Minulost byla zaměřena hlavně na strukturovaná data v relačních databázích, ale v současnosti tvoří nestrukturovaná data 80% množství ze všech dat. Například fotografie, videa, příspěvky na sociálních sítích, údaje z různých senzorů, audio soubory.

(Marr, 2015)

- **Veracity** (věrohodnost) odkazuje ke kvalitě a přesnosti dat, která je méně kontrolovatelná. Například příspěvky na sociálních sítích obsahují překlepy, hovorové výrazy či různé zkratky. Tento nedostatek do určité míry vynahrazuje velké množství dat.

(Marr, 2015)

- **Value** (hodnota). Schopnost proměnit data v užitečné (hodnotné) informace. Big data mohou přinést hodnotné informace do téměř jakékoliv oblasti obchodu nebo společnosti. Například pomáhají různým organizacím lépe porozumět a sloužit svým zákazníkům, optimalizovat své procesy, zvýšit bezpečnost nebo vylepšit péči o zdraví.

(Marr, 2015)

- **Variability** (proměnlivost) označuje údaje, jejichž význam se mění. Tento problém se vyskytuje například při strojovém zpracování lidské řeči, kdy mohou mít stejná slova různý význam podle kontextu. Variabilita bývá také chybně zaměňována s variety (různorodostí).

(Rijmenam, 2013)

- **Visualization** (vizualizace). Po zpracování dat je také důležité je vizualizovat takovým způsobem, který bude snadné číst a pochopit. Nejedná se zde o klasické koláčové grafy, ale o komplexní grafy zahrnující mnoho proměnných. Stále více firem se zabývá tímto problémem, a díky tomu bude pro organizace možné odpovědět na otázky, které je ani dříve nenapadly.

(Rijmenam, 2013)

4.3 Závislá a nezávislá data

Aggarwal (2015) popisuje rozdělení dat podle jejich vzájemné závislosti. Následující část vychází z tohoto zdroje. Z následujícího popisu je zřejmé, že závislost dat nemusí být vždy očekávaná nebo jasná (implicitní závislost). Při dalším dělení se některá data nedají striktně zařadit jen do jedné skupiny, protože jejich vlastnosti mohou splňovat podmínky pro více skupin.

Nezávisle orientovaná data

Tento druh dat je nejjednodušší, nejčastější a typicky odkazuje na multidimenzionální data. Příkladem může být tabulka v databázi, která obsahuje informace o zákaznících (jméno, věk, pohlaví, PSČ), což představuje jednotlivé dimenze a jednotlivé záznamy jsou nezávislé.

(Aggarwal, 2015)

Tabulka 1 Příklad nezávislých multidimenzionálních dat
Zdroj: zpracováno podle Aggarwala (2015)

Jméno	Věk	Pohlaví	PSČ
Kamil Novotný	28	M	39155
Eva Rychlá	23	Ž	50901

- **Kvantitativní multidimenzionální data** - sloupec Věk, v předchozí tabulce, představuje kvantitativní data. Někdy bývají označována jako numerická a mají tu vlastnost, že se dají přirozeně uspořádat. Pokud by ve všech sloupcích tabulky byla kvantitativní (numerická) data, jednalo by se o kvantitativní multidimenzionální data.

- **Kategorická a smíšená data** - datové sady často obsahují kategorické hodnoty, které jsou diskrétní a neuspořádané. Takové hodnoty jsou v tabulce obsaženy ve sloupcích Pohlaví a PSČ. Tento typ dat je označován jako kategorický, diskrétní nebo neuspořádaný. Hodnoty pohlaví jsou zvláštní v tom, že mohou obsahovat pouze dvě různé hodnoty, proto se dají označit za binární typ dat (viz dále). Díky tomu také mohou být zpracovány algoritmy pro numerická data. Data v tabulce mohou být celkově označena jako smíšená (mixed attribute), protože obsahují hodnoty kvantitativní i kategorické.
- **Binární a sadová data** - binární data mohou být speciálním případem multidimezionálních kvantitativních dat nebo multidimenzionálních kategorických dat. Kategorická data jsou binární v případě, že mohou nabývat jedné ze dvou diskrétních hodnot. Speciálním případem kvantitativních dat jsou proto, že mezi dvěma hodnotami existuje možnost řazení. Binární data mohou také reprezentovat sadová data, kde každý atribut slouží jako ukazatel, zdali je určitý prvek obsažen v seznamu či ne. Běžně se tento typ dat vyskytuje v případě nákupního košíku v internetovém obchodě, kdy například je dán seznam zboží *{chleba, vejce, sýr, jogurt, mléko}* a v binární reprezentaci *11010* ukazuje číslo 1, že výrobek je obsažen v košíku a číslo 0 naopak, že není obsažen.
- **Textová data** mohou být chápána, v závislosti na jejich reprezentaci, jako multidimenzionální data nebo řetězec (string). V případě řetězce se jedná o závisle orientovaná data, která jsou zmíněna níže. Protože je obtížné a neefektivní, při rozsáhlých aplikacích, vytvořit určité řazení jednotlivých slov, je místo řetězcové reprezentace v praxi často využívána reprezentace vektorová, která udává frekvenci výskytu slov. Pak se dají tato data uchopit jako kvantitativní multidimenzionální data, ale algoritmy pro jejich zpracování musí být většinou upraveny.

(Aggarwal, 2015)

Závisle orientovaná data

Z hlediska analýzy dat na sobě mohou být data určitým způsobem závislá, například prostorově, časově nebo prostřednictvím explicitního vztahu k síti. Díky těmto závislostem mají mezi sebou data určité vztahy, které mohou být využity pro získání nových informací. Závislost dat se dá rozdělit do dvou supin:

- **Implicitní závislost** není jasně daná. Může se objevit například v situaci, kdy se provádí analýza údajů z teplotního senzoru a po sobě jdoucí naměřené hodnoty jsou neobvykle rozdílné.
- **Explicitní závislost** se typicky objevuje v grafech nebo počítačových sítích, kdy hrany mezi dvěma vrcholy explicitně definují jejich vztah.

(Aggarwal, 2015)

Implicitně a explicitně závislá data se dají dále dělit a popsat mnohem podrobněji, ale není nutné v této práci zacházet do zbytečných podrobností.

4.4 Kvalitativní a kvantitativní data

Rudová (2001) rozděluje data do dvou tříd: kvalitativní a kvantitativní, ale kvalitativní data popisuje jen stručně. Zdroj Kvalitativní data (2014) o nich říká následující:

„Kvalitativní data jsou nečíselná data pomocí kterých popisujeme slovně (například spokojenost zákazníka, barvu, chuť, krásu). Popisují kvalitu věcí a jevů. Někdy se používá pojem měkká data. Jsou opakem kvantitativních dat. [...] Pomocí kvalitativních (měkkých) dat popisujeme vlastnosti, charakteristiku, pocity a další na rozdíl od kvantitativních dat, kterými definujeme a měříme. Měkká data mohou být pozorována, nevyjadřují se jako čísla. Mohou ale nabývat nominálních hodnot, jako je pohlaví, barva).“

(Kvalitativní data, 2014)

Dále Rudová (2001) rozděluje kvantitativní data na nominální, ordinální, intervalová a spojitá:

- **Nominální data** jsou číselná, reprezentují kategorie nebo atributy. Jejich důležitou vlastností je, že nemají relativní význam. Jinými slovy, pokud by mužské a ženské pohlaví bylo reprezentováno jako nominální data čísly 1 a 2: *„...i když muž = 1 a žena = 2, není relativní hodnota toho, že je někdo žena, dvakrát větší ani vůbec větší, než že je někdo muž.“*
- **Ordinální data** jsou čísla reprezentující kategorie, které mají, na rozdíl od nominálních dat, relativní význam. Využívají se k hodnocení významnosti nebo síly. Například výše rizika, spojeného s určitou investicí, může být hodnocena od 1 do 5.
- **Intervalová data**, stejně jako ordinální data, jsou číselná s relativním významem. Navíc nemají nulový bod a operace sčítání a odečítání zde mají smysl. *„Například řada finančních institucí používá hodnocení rizika s daleko jemnější definicí než pouhé hodnoty 1 až 5, jak tomu bylo v předchozím příkladě. Typický rozsah bývá od 300 do 800. Pak je možné porovnávat hodnocení měřením rozdílů.“*
- **Spojitá data** jsou nejčastější pro vytváření prediktivních modelů. Obchodní údaje, jako tržby, minuty, zůstatky, jsou spojitá data. Je možné s nimi provádět základní aritmetické operace (sčítání, odečítání, násobení, dělení).

(Rudová, 2001)

5 Zdroje dat a jejich integrace pro analýzu

Za zdroje dat mohou být označeny sociální sítě, zpravodajské servery, elektronická pošta nebo streamovací služby, které se dají zkrátka uvést jako WWW (World Wide Web). Dále například finanční transakce, zdravotní záznamy, dotazníková šetření a různá měření. V této práci se však zdroji dat myslí spíše systémy, které data uchovávají a pomáhají je získávat nebo vytvářet, ať už manuálně či automatizovaně. Jako zdroje dat se často udávají různé klientské databáze, aplikace či systémy, ze kterých se data následně sjednocují zpravidla v datových skladech (Data Warehouses). Datové sklady a jiné systémy mohou být také považovány za zdroj dat, pokud z nich čerpá další systém nebo člověk. Laursen a Thorlund (2010) hovoří o tom, že označení „zdroj dat" je poměrně obecné a spíše rozlišují mezi systémy, které data generují či negenerují. Na druhou stranu je zvláštní, že mezi systémy generující zdrojová data zařazují například webové logy, což jsou spíše datové soubory generované nějakým systémem. Navíc, datový sklad neřadí pod systémy generující data, ale samozřejmě i ten generuje logové soubory, které by mohly být předmětem zkoumání. Dalo by se říci, že tento log obsahuje metadata, ale je otázka, kdy metadata chápat jako „klasická" data a naopak. Záleží na perspektivě. Zdroje dat jsou tedy různé podnikové systémy či aplikace, výstupy z podnikových procesů a nemusí být primárně chápány jako zdroj dat, ale stávají se jím, pokud jsou z nich data čerpána. Podle Rudové (2001) se nemusí jednat jen o interní zdroje v rámci organizace, ale i o zdroje externí, kdy mohou být nakupována data o potenciálních zákaznících. Integrace dat má pak za úkol jednotlivé zdroje dat propojit a data sjednotit podle požadovaných kritérií do jednotného systému nebo pohledu, což mohou být datová tržiště (Data Marts) nebo, již zmíněné, datové sklady i jiné systémy, což usnadňuje provádění komplexních analýz a dotazů nad daty. Data mezi těmito podnikovými systémy mohou proudit oběma směry v závislosti na provedeném řešení, které bylo navrženo jako vhodné pro danou společnost. Jak píšou Pierson a Porway (2015), hodnotné informace mohou být získány i z jednoho zdroje dat, ale často se různé zdroje kombinují, aby bylo dosaženo širšího pohledu nebo pohledů z jiných dimenzí, což vede k lepším výsledkům.

5.1 Interní a externí zdroje dat

Zdrojů dat, přinášejících cenné informace, je mnoho. Organizace mohou využívat data z vlastních zdrojů (interní zdroje) nebo ze zdrojů třetích stran (externí zdroje). Následující popis je podle Rudové (2001).

Interní zdroje

Interní zdroje mají často nejvyšší prediktivní potenciál, protože se jedná o data specifické pro danou společnost. Patří sem **zákaznické databáze**, které obsahují identifikační a kontaktní informace o zákaznících, zakoupené produkty či služby, demografické údaje (pohlaví, věk, příjem), podrobnosti o poskytnutých nabídkách a podobně. **Transakční databáze** uchovávají informace o aktivitách zákazníka, typický příklad uchovávaných dat je ID zákazníka, číslo účtu, obchodní aktivita (jednotlivé transakce) a datum uskutečnění. Přehled o uskutečněných nabídkách, stávajícím či potencionálním zákazníkům, obsahuje **databáze historie nabídek**. Z velké části se její popis shoduje se zákaznickou databází. Odlišuje se například tím, že obsahuje i informace o potencionálních zákaznících a některými dalšími informacemi o nabídkách.

(Rudová, 2001)

Externí zdroje

Externími zdroji jsou myšleny hlavně informace od firem, které jsou uvedeny jako prodejci a kompilátoři seznamů. Jedná se o firmy, které sbírají, nakupují a prodávají informace o zákaznících dalším firmám. Ve většině případů se nejedná o jejich hlavní činnost, kterou je prodej zboží přes katalogy nebo časopisy. Zdroje dat mohou být jejich vlastní, ale může se jednat i o informace z registrací řidičských průkazů nebo telefonní seznamy. Tato data mohou být sjednocována a čištěna pro vyšší hodnotu. Kompilátoři seznamů mohou být více či méně zaměřeni na určitý druh dat nebo oblasti jejich prodeje.

(Rudová, 2001)

5.2 Zdrojové systémy a data a jejich využití

Laursen a Thorlund (2010) popisují některé systémy a typy zdrojových dat následovně:

- **Účtovací systémy** (billing systems) vystavují faktury zákazníkům. Analýzou těchto údajů se dá provést segmentace na základě ceny nebo druhu zboží.
- **Upozorňující systémy** (reminder systems) upozorňují zákazníky na nezaplacení jejich účtů. Na základě těchto dat může být zákazník například pokutován nebo naopak odměněn za vzorné platby.
- **Systémy na vymáhání dluhů** (debt collection systems) informují o stavu dlužících zákazníků, kteří byli převedeni k externím firmám na vymáhání pohledávek. Na základě těchto informací se může organizace rozhodnout, se kterými zákazníky nechce dál spolupracovat, dokud se nedosáhne vyrovnání.
- **Systémy pro řízení vztahu se zákazníky** (CRM - Customer Relationship Management). Tyto systémy obsahují údaje z různých volání a rozhovorů se zákazníky. Na základě jejich stížností se může organizace rozhodnout, co dělat lépe.
- **Informace o výrobcích a jejich spotřebě** poskytují přehled o kupovaném zboží, případně ho spojuje se zákazníky, kteří ho kupují.
- **Historie marketingových kampaní** umožňuje sledovat efektivitu marketingových iniciativ a reakce zákazníků obecně nebo i individuálně. Informace agregované z více různých kampaní ukazují, které druhy marketingu jsou nejpřínosnější.
- **Webové logy** odhalují chování zákazníků na webových stránkách. Nabízejí informace o jejich počtu a pohybu na webu. Při využití cookies nebo registrace zákazníků je možné toto sledovat i z pohledu konkrétního zákazníka ve spojení s účtovacími systémy nebo CRM systémy.
- **Informace o lidských zdrojích, jejich kompetence, mzdy, historie.** Tyto informace jsou často podceňovány ve větších společnostech. Umožňují zjistit zaměstnanecké absence a jejich důvody, kteří zaměstnanci jsou

obtížně udržitelní nebo prospěšní pro podnik. Informace o odpracovaných hodinách, v kombinaci s výstupy celopodnikových informačních systémů (respektive systémy pro podporu plánování, Enterprise Resource Planning - ERP), mohou poskytnout klíčové ukazatele výkonnosti a produktivity (KPIs - Key Performace Indicators).

- **KPIs** se využívají při monitorování aktuálních procesů, ale při jejich nahromadění se dají využít pro optimalizaci procesů, protože odhalují korelace mezi aktivitami a finančními zisky organizace.
- **Data mining výsledky**. *„Tyto výsledky, které mohou být segmentace, přidané prodejní modely nebo věrnostní segmentace, poskytují historii, když jsou uloženy v datových skladech. Jako v případě KPI, mohou být tyto informace využity pro získání vědomostí o kauzálních vztazích napříč různými kampaněmi a uvést tak tržní mechanizmy v širším kontextu.“*
- **ERP systémy** obsahují informace ze systémů pro správu účetnictví (Accounting Managemet Systems), kde jsou zachyceny finanční transakce v účetním formátu. *„Mohou se vztahovat k informacím KPI, pokud chceme odhalit korelace mezi iniciativami a zdali jsou výsledky podle očekávání.“*

(Laursen a Thorlund, 2010)

5.3 Pravidla a problémy při integraci dat

Při integraci dat („tradičních“ i rozsáhlých) udávají Hurwitzová et al. (2013) tři základní pravidla:

- **Udržovat kvalitu dat podle perspektivy.** Důraz na kvalitu dat by se měl odvíjet podle fáze analýzy, která bude prováděna. Při počátečním zpracování velkého objemu dat je těžké kontrolovat jejich kvalitu. Postupně, když se z tohoto množství údajů vybere podmnožina, která má smysl pro organizaci, se klade na kvalitu dat větší důraz. Dále je kvalita důležitá, pokud mají výsledky korespondovat s kontextem historických údajů organizace. Čím více je rozhodování společnosti závislé na výsledcích těchto analýz, tím více je kvalita rozhodujícím faktorem mezi úspěchem a selháním.

- **Zvážit požadavky na zpracovávání dat v reálném čase.** Pokud je pro organizaci důležité analyzovat přicházející data co nejrychleji, je nutné vědět, jak integrovat tento proud dat (streaming data) do prostředí organizace pro prediktivní analýzy.
- **Nevytvářet zbytečné zdroje dat.** Data, která bude organizace zachycovat, by jí měla být prospěšná, v souladu s jejími cíly a technickými možnostmi.

(Hurwitzová et al., 2013)

Čech a Bureš (2009) shrnují některé problémy při integraci dat. Jedná se například o redundanci, kdy v různých podnikových databázích mohou být o stejném zákazníkovi odlišné nebo jinou formou zapsané údaje, dále odlišné formáty dat (míněno množné číslo od slova datum) v různých státech nebo sjednocení měn, nevyplněné hodnoty nebo reference na již neexistující (smazaný) záznam.

5.4 Systémy pro integraci dat

Všechna potřebná data, vygenerovaná z různých oddělení, poboček nebo aplikací organizace, by měla být sjednocena v cílové databázi. To může být například datový sklad (Data Warehouse), datové tržiště (Data Mart), provozní datové úložiště (Operational Data Store) nebo hybridní řešení. Tato úložiště by měla být postavena tak, aby nad daty bylo možné provádět různé komplexní dotazy a reporty, nebo aby se údaje daly využít pro jiné analytické nástroje.

(Scheps, 2008)

- **Datový sklad (DWH - Data Warehouse)** sjednocuje data organizace z různých zdrojů. Obsahuje historické údaje, takže je možné porovnávat stejný druh dat z různého časového období a provádět mnoho jiných analytických operací, které slouží pro podporu rozhodování organizace.

  (Čech a Bureš, 2009)

  Datový sklad nemusí být nutně jedna velká databáze všech dat, může se jednat o více propojených systémů s jednotným přístupem.

  (Scheps, 2008)

- **Datové tržiště (DMA - Data Mart)** je v principu typ datového skladu, který však operuje v menším měřítku, například v jednom firemním oddělení nebo pobočce.

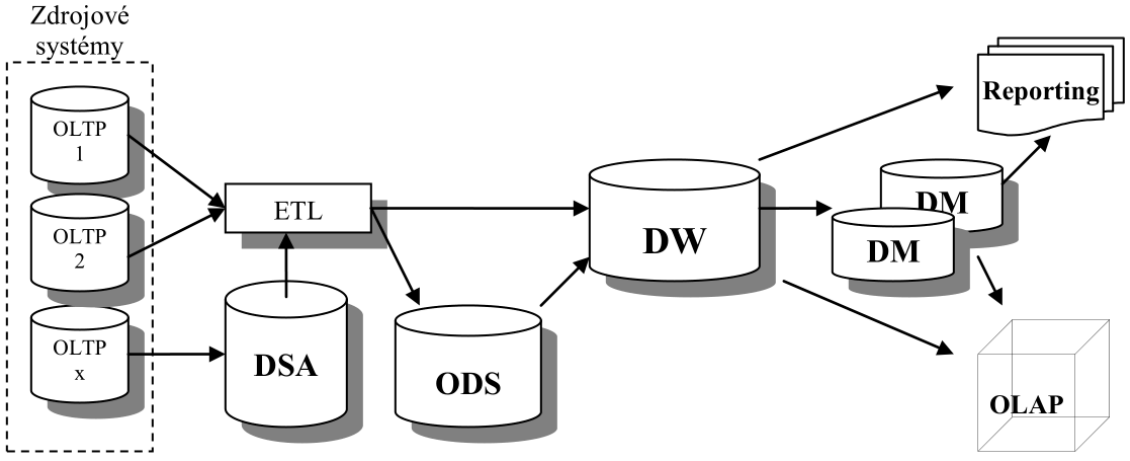
(Pour, Maryška a Novotný 2012)

- **Provozní datové úložiště (ODS - Operational Data Store)**
  „Oproti datovému skladu se data v ODS mění (v reálném nebo skoro reálném čase) a ODS tak může sloužit k rychlým a aktuálním analýzám okamžitého vývoje napříč daty z více aplikací. V ODS jsou data v detailní atomické podobě na rozdíl od datového skladu, kde se může používat již určitých agregací.“

(Čech a Bureš, 2009)

Velmi důležitou částí při řešení datových skladů je právě samotný přenos dat. Tento proces je označován jako **ETL (Extract, Transform, Load)**. Při navrhování datových skladů může příprava nástrojů pro ETL zabrat kolem 70 % času. Úkolem ETL je extrakce dat ze zdrojových databází, transformace dat do požadované konzistentní podoby a jejich transport do datových skladů.

(Kimball a Caserta, 2004)



Obrázek 1 Řešení architektury datového skladu
Zdroj: Čech a Bureš (2009)

Obrázek výše zachycuje jedno z možných řešení při návrhu datových skladů. Obsahuje některé nepopsané systémy:

- **OLTP (Online Transaction Processing)** jsou transakční systémy, vyznačují se vysokou propustností, rychlou odezvou a umožňují přístup mnoha uživatelům najednou. Obsahují aktuální data.

  (Özsu a Valduriez, 2011)
- **OLAP (Online Analytical Processing)** systémy slouží pro analýzu sumarizovaných historických dat, většinou z datových skladů, která mohou přicházet z několika operativních databází.

  (Özsu a Valduriez, 2011)
- **DSA (Data Staging Area)** je dočasné úložiště dat, které přijímá data z produkčních (zdrojových) databází. Data jsou detailní, neagregovaná, bez časové dimenze a často nekonzistentní. Po jejich zpracování se data přesouvají do datového skladu nebo tržiště a z úložiště DSA se odstraní.

  (Pour, Maryška a Novotný 2012)
- **Reporting** - tyto aplikace se řadí ke klientským aplikacím. Jako zdroj dat mohou využívat datový sklad, transakční nebo OLAP databáze, ze kterých získáme požadované informace pomocí dotazů.

  (Pour, Maryška a Novotný 2012)

Výše uvedená architektura není jedinou možnou nebo správnou. Samozřejmě záleží na specifických potřebách organizace. Zdroje například nemusí být jen transakční systémy, datová tržiště nemusí čerpat jen z datového skladu, nemusí být potřebné dočasné úložiště nebo mohou být využity i jiné druhy systémů.

6 Příprava dat

„Příprava dat je v procesu vytváření modelu jedním z nejdůležitějších kroků. Od nejjednodušší analýzy až po nejsložitější model je kvalita vstupních dat klíčem k úspěchu projektu. Slavný výrok ‚Garbage in, Garbage out! (Odpadky na vstupu = brak na výstupu)' je pro tento případ velmi přiléhavý."

(Rudová, 2001)

6.1 Kvalita dat

„...je zřejmé, že management kvality dat je proces, který vede celým životním cyklem dat. Z organizačního hlediska by měl být považován za hlavní cíl společnosti v příslušném organizačním rámci."

(Grossmann a Rinderle-Ma, 2015)

Zvyšování kvality dat je také označováno jako **čištění dat**. Kvalita dat se dá hodnotit podle sedmi kvalitativních dimenzí: relevance, přesnosti, kompletnosti, dochvilnosti, konzistence, koherence a spolehlivosti. **Relevance** udává míru významnosti dat pro řešenou otázku. **Přesnost** říká, jak moc data odpovídají skutečnosti. Důvodem nepřesností mohou být například překlepy, chybějící hodnoty nebo nesprávná měření. **Kompletnost** se řeší ve vztahu k populaci, kterou mají data popisovat. V případě různých průzkumů je proto velmi důležité určit, jak bude průzkum prováděn. Častou chybou jsou nesprávně nastavená časová měřítka, kdy mohou například jako nejmenší měrnou jednotku rozlišovat jednotlivé dny, ale už nerozlišují časové uspořádání událostí během jednoho dne. Kompletnost je míra úplnosti dat s ohledem na hloubku, šířku a rozsah. **Dochvilnost** znamená, že vybraná data by měla reprezentovat aktuální stav. Problém může nastat v případě údajů z externích zdrojů, kdy nemůže být jejich aktuálnost zaručena. **Konzistence** je popisována jako stupeň ekvivalence dat v redundantních nebo distribuovaných databázích. **Koherencí** je myšlena soudržnost nebo souvislost dat při jejich kombinování různými způsoby a za různými účely. Konzistence a koherence mají nejvyšší význam v případě kombinování více zdrojů dat, kdy mohou být problémem například různé hodnoty u stejných atributů v různých databázích. *„**Spolehlivost** je charakteristika*

informační infrastruktury pro ukládání a načítání informací přístupným, bezpečným, udržovatelným a rychlým způsobem.“

(Grossmann a Rinderle-Ma, 2015)

6.2 Problémy se zdrojovými daty

I přes veškeré úsilí a návrhy řešení, která se budou snažit udržet nejvyšší kvalitu dat, se díky komplexitě a množství faktorů, které mohou výsledek ovlivnit, nejspíš nepodaří zcela vyhnout následujícím problémům.

Chybějící hodnoty (missing values)

Jeden z nejčastějších problémů jsou chybějící údaje. Data mohou chybět z různých příčin. Anderson (2015) je rozděluje do dvou typů a jako první udává **nonresponse** (bez odpovědi), což označuje chybějící pozorování, která nebyla vůbec zaznamenána. Příkladem jsou výzkumy formou dotazníků, jak vysvětluje Skalská (2010):

„...když se určitý podíl dotázaných osob vůbec nevyjádří v elektronickém formuláři ke spokojenosti se službou kvůli obavám, že kritické vyjádření by je mohlo v budoucnu poškodit (nedůvěřují například, že odpovědi jsou anonymní), pak získané odpovědi mohou být všechny pochvalné, ale celkové závěry budou zkreslené v důsledku chybějících dat.“

(Skalská, 2010)

Druhý případ chybějících hodnot je označen jako **missingness** (absence, chybění), kdy nejsou pozorování přítomna v datové sadě z příčiny technických nebo lidských chyb, nebo se jedná o náhodu. Anderson (2015) missingness dále rozděluje:

- **Missing completely at random (MCAR)** - data chybějí kompletně náhodně, když není nalezen žádný vzorec, podle kterého by se dala určit příčina. V takovém případě je pravděpodobnost chybějících hodnot stejná pro všechna pozorování. Jako příklad je uvedena cena akcií společnosti Apple. Čas je nezávislá proměnná X a cena je závislá proměnná Y. Cena je

funkcí času. Protože se chybějící hodnoty ceny (Y) žádným způsobem nevztahují k hodnotám X ani Y, jedná se o MCAR.

- **Missing at random (MAR)** - je nalezena určitá pravidelnost. Pokud by se našla v předchozím příkladě souvislost mezi chybějícími hodnotami Y (cena) a X (čas) a zároveň by nebyla nalezena souvislost mezi cenami samotnými, jednalo by se o MAR (hodnoty by například chyběly každé pondělí).
- **Missing not at random (MNAR)** - nejedná se o náhodu. Chybějící Y je funkcí k hodnotám Y anebo X. Ceny akcií chybí například každý první den v měsíci a chybí jen nejvyšší hodnoty.

(Anderson, 2015)

Řešení pro chybějící hodnoty

Anderson (2015) vysvětluje následující techniky pro řešení problému chybějících hodnot. V případě chybějících hodnot MCAR je jedním z řešení vymazat kompletně záznamy, ve kterých některé hodnoty chybí a pracovat jen s úplnými daty. Předpokladem je, že smazaných dat bude relativně malé množství a zůstane dostatek dat pro podání nezkreslených výsledků. Tato metoda je označena **casewise deletion** (také často označovaná listwise deletion nebo complete-case analysis). Druhou možností je metoda **pairwise deletion** (available-case analysis), kdy se z neúplných záznamů odstraní jen příslušné chybějící hodnoty proměnných a analýza probíhá s ostatními proměnnými v záznamu, které nechybí. Kromě mazání se využívají techniky **imputace** (imputation techniques), kdy se chybějící hodnoty proměnných odhadnou nebo přisoudí na základě známých dat. Dají se rozdělit na jednoduché a vícenásobné imputace.

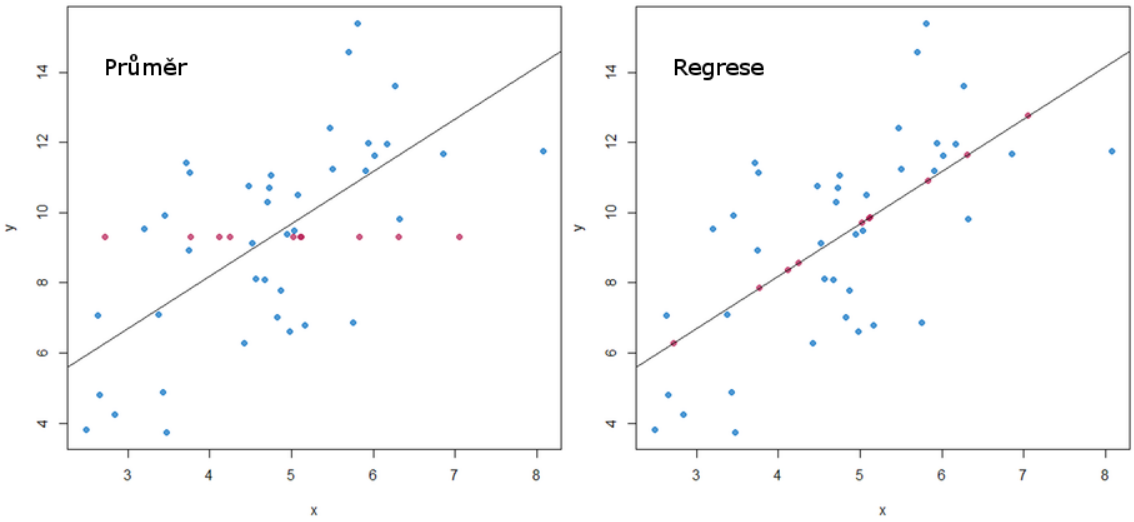
(Anderson, 2015)

Jednoduché imputace zahrnují tři techniky. Nejjednodušší z nich je **průměrová substituce**, kdy se chybějící hodnoty nahradí průměrnou hodnotou ze známých hodnot proměnných. Tato technika není příliš vhodná, zvláště v případě nahrazení většího množství hodnot. Vytrácí se tak variabilita v datech, která je

v podstatě klíčová pro statistické analýzy. Další metodou je **regresní substituce**, která se využívá při zpracování dat splňujících podmínky MCAR nebo MAR.

(Anderson, 2015)

Na následujícím obrázku je zachycen rozdíl mezi průměrovou a regresní substitucí, modré tečky reprezentují známé hodnoty, červené tečky představují hodnoty dopočítané imputačními technikami:



Obrázek 2 Srovnání průměrové a regresní substituce
Zdroj: Single imputation methods [2014], upraveno

Třetí jednoduchou imputací je **hotdecking**. Pro ukázku je dána následující tabulka:

Tabulka 2 Příklad dat pro hotdecking
Zdroj: zpracováno podle Andersona (2015)

X	Y
8	14
9	15
5	9
8	???

V tabulce chybí poslední hodnota pro Y. Za předpokladu, že spolu hodnoty proměnných X a Y souvisejí, se na místo neznámé přiřadí číslo 14, protože X má zde hodnotu 8, stejně jako v prvním řádku. Hotdecking se využívá v případě dat MCAR nebo MAR.

(Anderson, 2015)

Vícenásobné imputace spočívají v provádění množství iterací s využitím metody Monte Carlo, která vygeneruje pro chybějící hodnoty náhodná čísla na základě zvolených pravděpodobnostních funkcí a následně je provedena analýza. S každou další iterací by měly hodnoty proměnných konvergovat ke korektním hodnotám. Po sérii iterací jsou výsledky ze všech kroků zkombinovány do finální podoby. Tato technika je vhodná pro data, která mají normální rozdělení.

(Anderson, 2015)

Výčet technik a možností pro řešení chybějících dat není zdaleka kompletní a lze ho označit spíše za orientační.

Duplicitní záznamy

Uložené záznamy se mohou lišit podle jednotlivých systémů nebo typu záznamu. Například přehled investičního účtu je spojen s číslem účtu, přehled portfolia se dá uložit na individuální nebo domácnostní úrovni a obchodní historie těchto účtů je rozlišována na úrovni jednotlivých transakcí. Proto je důležité správně rozlišit, co jednoznačně identifikuje jednotlivé záznamy v souboru. Pokud jde například o soubor záznamů transakcí, bude obsahovat duplikáty čísel účtů nebo identifikátorů domácností. Toto je třeba mít na paměti. Mohlo by se jinak dosáhnout nechtěných výsledků například v případě otázky, kolik účtů spadá do jedné domácnosti. Většina systémů má vestavěné mechanismy pro sjednocování nebo seskupování záznamů podle zvolených kritérií, například příkaz GROUP BY v databázových systémech.

(Anderson, 2015)

Odlehlé hodnoty

Dalším problémem při analýze dat jsou odlehlé hodnoty (outliers). Takové hodnoty jsou významně vyšší či nižší od zbytku dat a mohou se objevit ve všech oblastech života. Za odlehlé lze považovat například stoletého člověka, osobu měřící přes 2 metry nebo domácnost s ročním příjmem sto milionů. Outliers mohou být důvodem ke zvážení spolehlivosti zdrojových dat. Některé statistické metody mohou být významně ovlivněny odlehlými hodnotami (například aritmetický průměr), a je proto vhodnější zvolit metody, které se označují jako

robustní a odlehlé hodnoty na ně nemají vliv (například modus nebo medián). Objevit odlehlé hodnoty je možné například za pomoci různých grafů (histogramy, krabicové grafy, kvantilové grafy) nebo testováním statistických hypotéz. Na otázku, jak se vypořádat s odlehlými hodnotami, není jednoznačná odpověď. Záleží také na problémové doméně a znalostech analytika. Jednou z možností je chybná data opět vymazat. Další variantou je provést analýzu s odlehlými hodnotami i bez nich a výsledky porovnat. V některých případech může být šance chybné hodnoty opravit.

(Anderson, 2015)

6.3 Transformace dat

Před pokročilými statistickými procedurami je často žádoucí data standardizovat. Se standardizovanými daty se lépe pracuje, mohou také zlepšit přesnost modelů a zjednodušit jejich interpretaci.

(Anderson, 2015)

Percentily

Transformování hodnot na percentily rozdělí statistický soubor na setiny. Například, když je určitá hodnota v 90. percentilu, znamená to, že 90 % hodnot je na stejné nebo nižší úrovni. Soubor může být rozdělen do jiných typů kvantilů, než jsou percentily. Často se využívají také kvartily, což je rozdělení na 4 části, nebo decily, které soubor rozdělí na 10 částí. Tyto transformace jsou vhodné v případě, že jsou data široce rozptýlena. Jako příklad jsou uvedeny příjmy v USA. Pokud model bude pracovat s hodnotami přímo v dolarech, důležité variace mezi nižšími příjmy (například pod $50,000), nebudou v porovnání s obrovskými variacemi na horní hranici (například $500,000, $100,000,000 a více) zřetelné. *„$10,000 je o hodně větší rozdíl pro někoho na 10. percentilu než pro někoho na 90.“* Proto je, pro zachycení variací na spodní hranici, lepší využívat percentily.

(Anderson, 2015)

Standardizované skóre

Standardní skóre se vypočítá odečtením střední hodnoty z jednotlivých pozorování a vydělením směrodatnou odchylkou. Ve většině případů leží hodnota

standardních skórů mezi -3 a 3. Tato metoda je vhodná pro prediktivní modely, které pracují s proměnnými velkého rozptylu (například hodnoty od 0 do 10 a zároveň hodnoty ve stovkách milionů). Tímto transformováním se hodnotám přidělí určitá váha s ohledem na jejich variace a umožní jasnější pohled na významnost jednotlivých proměnných a usnadní interpretaci.

(Anderson, 2015)

Dummy proměnné

Dummy proměnné jsou logické proměnné (také označované Boolean podle George Boolea, v knize od Rudové (2001) zase jako indikátorové), které nabývají jeden ze dvou stavů: pravda (1) nebo nepravda (0). Využívají se například v případě nečíselných hodnot, kdy chceme zjistit jejich výskyt. Například při prodeji třech příchutí zmrzliny (vanilková, čokoládová, jahodová) se vytvoří tři dummy proměnné „vanilka“, „čokoláda“, „jahoda“. Pokud se příchuť vyskytuje v objednávce, nastaví se příslušná hodnota na 1. Počet zvolených proměnných také závisí na tom, zdali musí být jedna z podmínek vždy splněna či nikoliv. Jestliže objednávka nemusí obsahovat ani jednu zmrzlinu, jsou nutné 3 proměnné, které budou v takovém případě nastaveny na 0. Počet dummy proměnných zastupujících dny v týdnu, musí být šest. Jedna pro každý den třeba od pondělí do soboty, a když jsou všechny hodnoty 0, jedná se o neděli. V tomto případě je jedna z podmínek vždy splněna, nemůže se stát, že by nebyl žádný den v týdnu. Tím se zabraňuje redundanci, která může být problematická pro prediktivní modely. Tento problém, multikolinearita, nastává při vícenásobné lineární regresi, kdy jsou proměnné na sobě lineárně závislé (hodnota jedné proměnné se dá odvodit z ostatních) a model je potom nepřesný.

(Anderson, 2015)

6.4 Data mining

Jedním z hlavních účelů, za kterým probíhají celé procesy sběru, udržování a přípravy dat, je provádění následných analýz pro získání hodnotných informací pro společnost, na jejichž základě provádí další rozhodování. V těchto

souvislostech se často hovoří o procesu data mining (DM). Do češtiny se DM překládá jako „dolování dat“ nebo „vytěžování znalostí z dat“ a podobně.

„Data mining vznikl jako samostatný obor, který je zhruba od konce 80. let 20. století řazen jako součást procesu vyhledávání znalostí z dat, KDD, Knowledge discovery from databases. KDD je oblast managementu znalostí, která se zabývá procesy automatizace vyhledávání znalostí z databází, formalizací výsledků, analýzy dat, modelováním dat i prezentací výsledků.“

(Skalská, 2010)

Jak popisuje Skalská (2010), existují více či méně odlišné definice procesu DM, ale shodují se v určitých bodech. Jedná se o výběr dat z databází, přípravu a transformaci dat, analýzu a prezentaci či interpretaci výsledků. Někdy je tento popis spíše přisuzován zmíněnému KDD, pod nějž DM spadá. Při DM jsou využívány informační technologie a různé analytické postupy, vycházející převážně ze statistiky. DM přináší prospěch do nejrůznějších oblastí lidské činnosti. Ve spojení s DM je možné narazit na pojmy, jako umělá inteligence, strojové učení nebo neuronové sítě. Důvodem může být snaha o nahrazení lidské složky, která zde hraje důležitou roli. Pro řešenou úlohu může být nutné mít o související oblasti určité znalosti, využívat zkušeností nebo intuici. Člověk je schopen abstraktního myšlení, což dovoluje zobecňovat modely pro DM nebo postupovat analogicky v podobných situacích. Dalším důvodem může být například snaha o automatizaci zpracování nestrukturovaných dat. Algoritmy s prvky umělé inteligence také řeší vyhledávání souvislostí v datech, které nejsou známé a očekávané. DM lze rozdělit na explorační a prediktivní. **Explorační DM** přináší výsledky popisného charakteru, které jsou vhodné pro souhrny a vizualizace dat. **Prediktivní DM** má za cíl předpovědět budoucí vývoj, vycházející z analýzy historických dat.

(Skalská, 2010)

7 Business inteligence, možnosti vybraného SW

Business intelligence (BI) patří k těm případům, které není dobré odbýt jednou definicí. Různí experti a firmy si definice mnohdy přizpůsobují tak, aby to vyhovovalo jejich specializaci. Zároveň se také BI vyvíjí s dobou a s dostupnými prostředky. Grossmann a Rinderle-Maová (2015) poukazují na článek z roku 1958 „A Business Intelligence System“ (Luhn, 1958), kde je tento systém představen: *„Automatický systém vyvíjený pro šíření informací do různých částí průmyslové, vědecké nebo státní organizace“*. Definice bývají také dosti obecné nebo pojaté příliš ze široka, čemuž se ale v případě BI ani nedá docela vyhnout. Stručně, do jedné věty, by se dal význam BI shrnout například takto: „Dělat vše pro možnost lepšího rozhodování v byznysu.“ Za slovem „vše“ se může skrývat opravdu cokoliv, jak uvádí Scheps (2008), který se zabývá problémem podobně:

„Ať už voláte psychickou horkou linku, máte armádu poradců nebo hradby z počítačů, ve kterých víří vaše data; pokud vám to pomůže lépe zvládnout současnou situaci a poskytne vhled do toho, co dělat v budoucnu, je to BI.“

(Scheps, 2008)

Scheps (2008) také popisuje BI jako jakoukoliv aktivitu, nástroj nebo proces, který opatří nejlepší informace pro podporu procesu rozhodování. V podobném duchu je i následující definice:

„Business intelligence (BI) 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.“

(Pour, Maryška a Novotný 2012)

Grossmann a Rinderle-Maová (2015) vybrali některé charakteristiky BI, které se shodují v různých definicích:

- Hlavním úkolem BI je podpora rozhodování pro určité cíle v kontextu podnikových aktivit v různých oblastech.
- Podpora rozhodování vychází z empirických dat, různých znalostí a teorií pro sběr informací.
- Podpora rozhodování je realizována jako systém využívající možnosti informačních a komunikačních technologií.
- BI systém dodává informace ve správný čas, správným lidem a ve správné formě.

(Grossmann a Rinderle-Maová, 2015)

Grossmann a Rinderle-Maová (2015) také hovoří o různých fázích vývoje. Proto bývá BI (nebo bývalo v dřívějších fázích vývoje) také chápáno ve smyslu větší orientace na data a jejich analýzu. Mezi BI nástroje se řadily především systémy pro podporu rozhodování (DSS - Decision Support Systems) nebo datové sklady ve spojení s OLAP systémy, nebo se s BI ani nespojovaly, dokud nepřešlo do širšího povědomí. Tato tvrzení podpírají i následující citace:

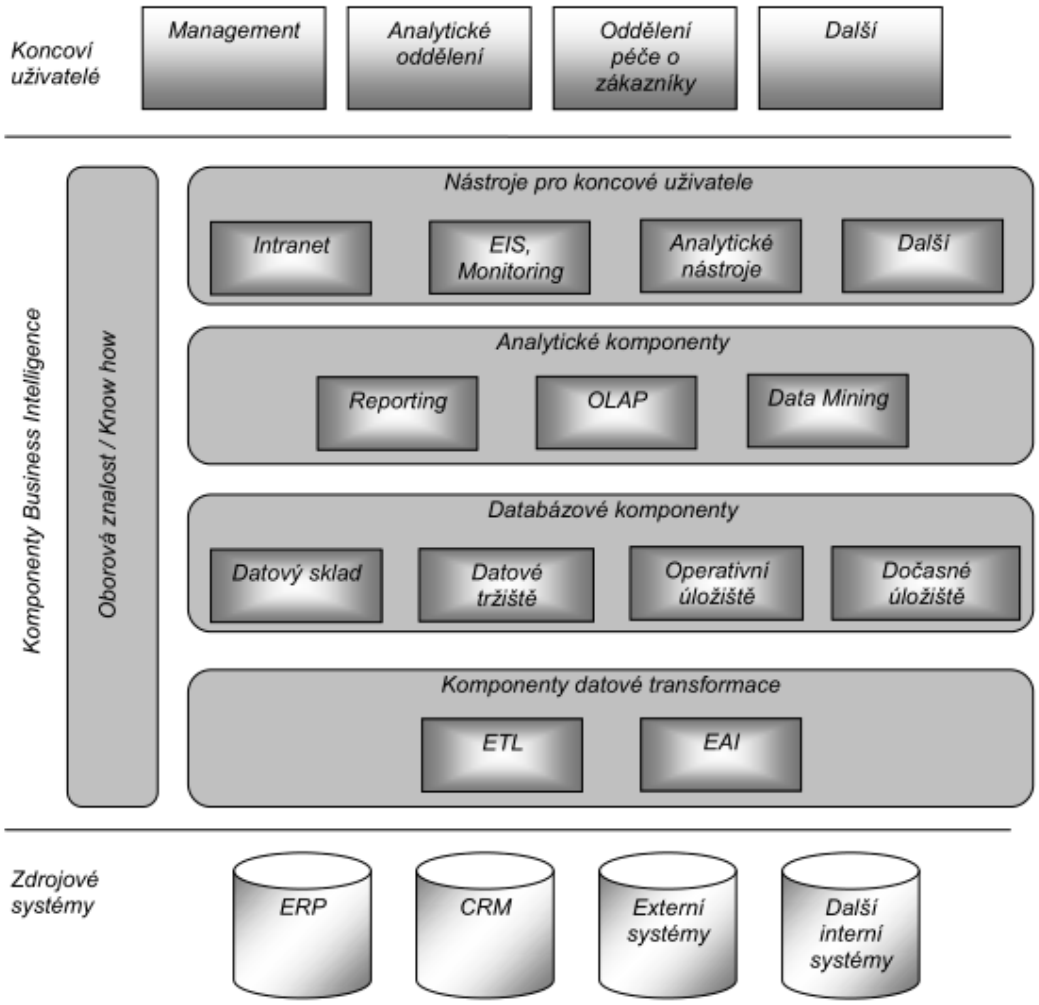
„Pod ‚úderným' názvem se skrývá především OLAP řešení, někdy také včetně datových skladů. Pod tento název také bývají zařazovány nástroje pro dolování v datech."

(Čech a Bureš, 2009)

„Pro aplikační oblasti, spojené zejména s podnikatelskou sférou, se proces data mining a vyhledávání znalostí z dat označuje business intelligence (BI). Proces je navržený a implementovaný tak, aby jeho výsledky byly interpretovatelné a přímo využitelné pro podnikatelské rozhodování. Tyto směry využití se někdy označují DM-BI."

(Skalská, 2010)

Čech a Bureš (2009) zachycují obecnou koncepci architektury BI na následujícím obrázku:



Obrázek 3 Obecná koncepce architektury BI
Zdroj: Čech a Bureš (2009)

7.1 IBM SPSS Modeler

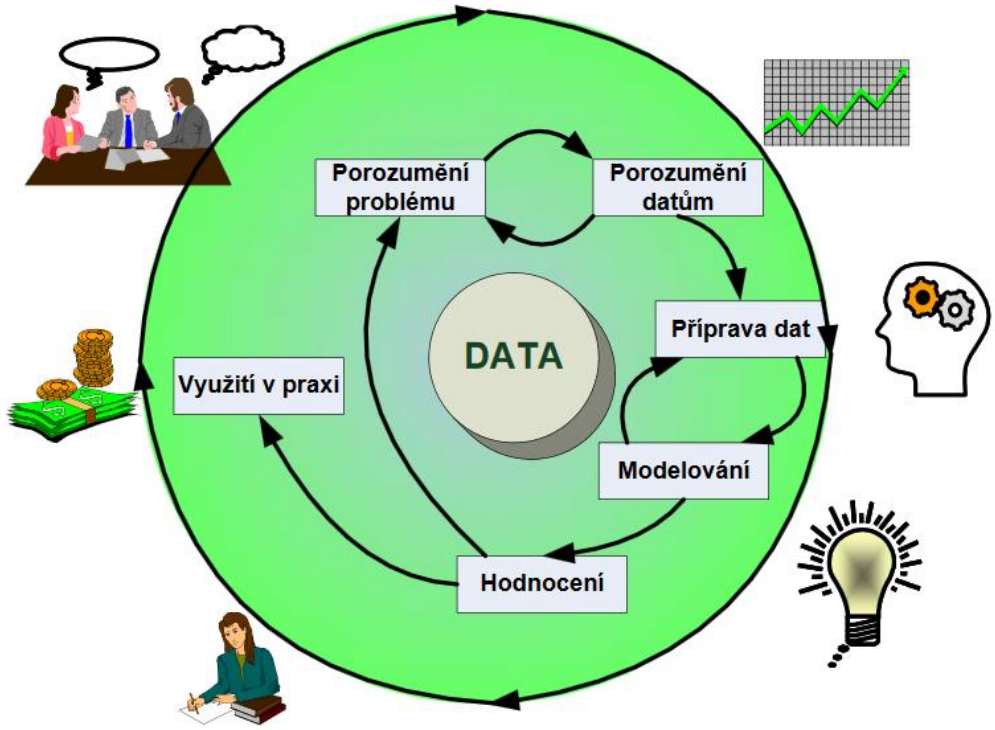
SPSS Statistics je modulově navržený statistický software, jehož první verze vznikla v roce 1968 pro sálové počítače. Je v hodný pro zpracování relativně rozsáhlého množství dat (stovky tisíc pozorování, každé popsané desítkami proměnných). Později byl odkoupen firmou IBM a přejmenován na IBM SPSS Statistics. **IBM SPSS Modeler** (původní název Clementine pod firmou ISL) slouží pro datové modelování a při tom využívá IBM SPSS Statistics. Jedná se o komerční software pro data mining.

(Skalská, 2010)

Modeler podporuje jednu z nejpoužívanějších metodologií pro data mining CRISP-DM (Cross Industry Standard Process for Data Mining), jejíž životní cyklus je

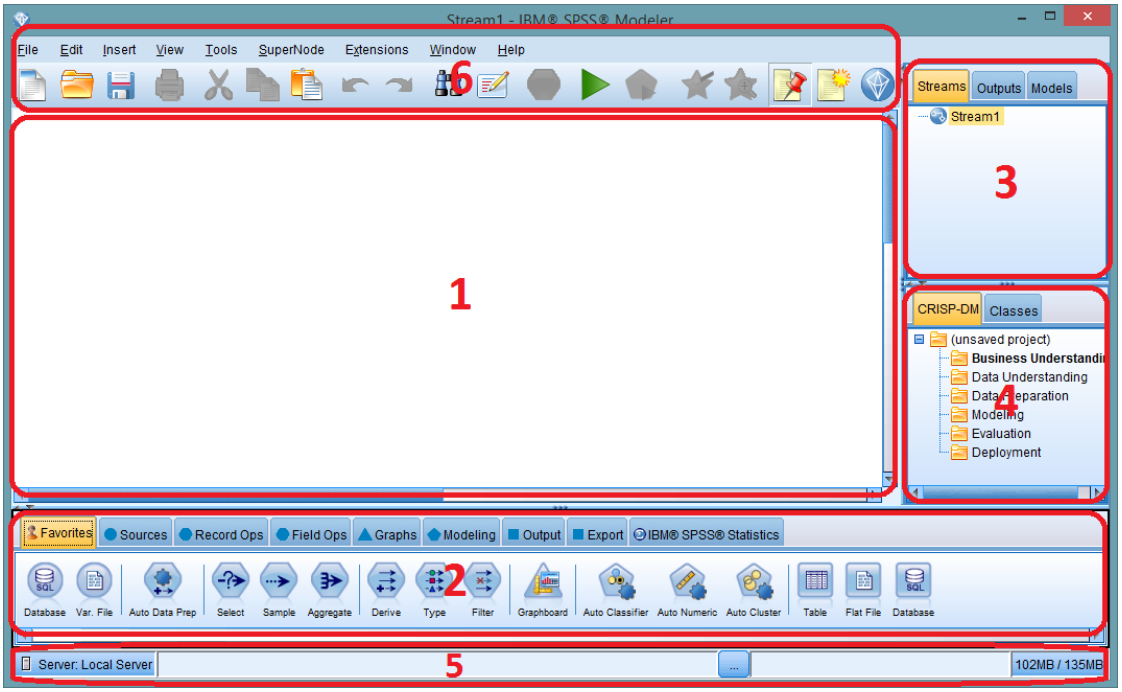
znázorněn na obrázku níže. Jednotlivé kroky nejsou pevně dány, mohou se navzájem zpětně ovlivňovat a je třeba se k nim vracet. Modeler je vhodný pro úlohy, jako detekce a predikce podvodů, predikce odcházejících zákazníků, odhalování skrytých vazeb a struktur, predikce budoucích trendů a další. S příslušnými moduly podporuje například skriptování v jazyce R, znalost programování však není podmínkou pro využívání tohoto softwaru, který nabízí přehledné grafické uživatelské rozhraní s poměrně jednoduchým ovládáním. Nabízí různé algoritmy pro seskupování, klasifikaci, asociaci a predikci (například C5.0, Decision List, TwoStep, Apriori, Carma a další). Dokáže pracovat s daty z různých zdrojů a v různých formátech, strukturovanými i nestrukturovanými a kombinovat je. Program obsahuje rozsáhlou nápovědu včetně ukázkových příkladů využití.

(Petr, 2012)



Obrázek 4 Životní cyklus metodologie CRISP-DM
Zdroj: Petr (2012)

Před ukázkou aplikace ještě následuje stručný popis základního uživatelského rozhraní programu (verze 18) podle obrázku:



Obrázek 5 Uživatelské rozhraní IBM SPSS Modeler

1. **Plátno** (stream canvas) je hlavní pracovní plochou programu, kde se tvoří samotný stream v podobě grafu. Každá operace je reprezentována příslušným uzlem (ikonou), které jsou propojeny hranami ve tvaru šipek podle směru proudění dat.
2. **Paleta uzlů** obsahuje většinu nástrojů (uzlů) pro modelování, které jsou zde seskupeny pod záložkami podle typu.
3. **Manažeři** - okno obsahující tři záložky pro správu jednotlivých streamů, výstupů a modelů.
4. **Projekty** - okno pro data mining projekty (skupina souborů potřebných k úloze). Nabízí rozdělení podle metodologie CRISP-DM nebo kategoricky podle tříd vytvořených objektů (data, streamy, modely).
5. **Informační** a **stavový řádek**, který zobrazuje průběh jednotlivých operací a jiné informace.
6. Základní **nabídkové menu** a **nástrojová lišta**.

(IBM®, 2016)

Ukázka aplikace

Součástí softwaru IBM SPSS Modeler je dokumentace (nápověda), která obsahuje i ukázky řešených příkladů. Z těchto ukázek je zde vybrán příklad klasifikace buněčných vzorků s použitím techniky SVM (Support Vector Machine).

SVM je klasifikační a regresní technika, která je vhodná pro rozsáhlé datové sady (s velkým množstvím prediktorů).

(IBM®, 2016)

Zadání: Na základě dostupných dat, získaných ze vzorků lidských buněk od pacientů s rizikem rakoviny, vytvořit SVM model, který zpracuje data budoucích pacientů a pokusí se předpovědět, zdali bude nádor benigní (nezhoubný) nebo maligní (zhoubný). Předchozí analýza dat totiž ukázala, že se charakteristiky pro benigní a maligní nádory významně liší.

(IBM®, 2016)

Použitá data jsou reálná a vycházejí ze zjištění Dr. Williama H. Wolberga. Údaje jsou dostupné online z UCI Machine Learning Repository a také jsou součástí instalace IBM SPSS Modeler. Konkrétně zde bude použit soubor obsažený v instalaci, který je navíc doplněn o první řádek s názvy jednotlivých proměnných. Pro každé pozorování je zde 11 proměnných (prediktorů), což je relativně malé množství. Proměnné jsou představeny v tabulce níže.

Sada obsahuje 699 případů od ledna 1989 do listopadu 1991. Data vycházejí z výzkumu rakoviny prsu. Hodnoty všech atributů jsou v číselné formě. *ID* je identifikační číslo záznamu, *Class* nabývá hodnoty 2 (benigní) nebo 4 (maligní). Ostatní proměnné nabývají hodnot 1 (nejblíže k benignímu stavu) až 10 (neblíže k malignímu stavu).

(Breast Cancer Wisconsin (Original) Data Set, 1992)

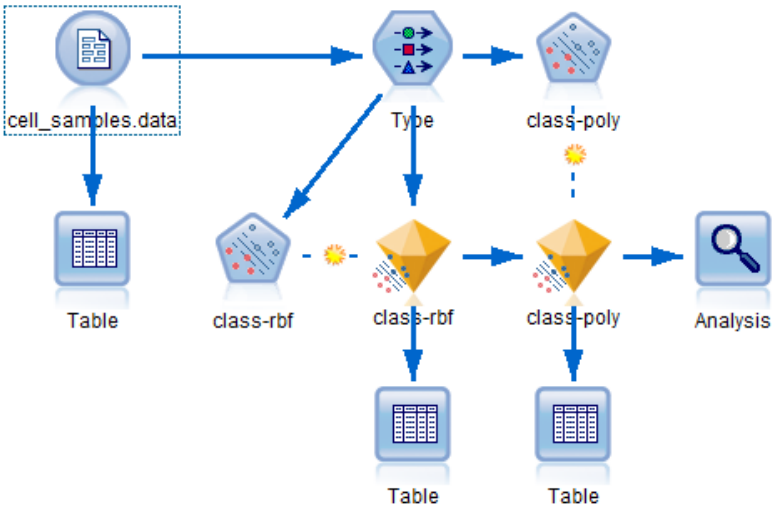
Tabulka 3 Přehled proměnných pro každé pozorování
Zdroj: IBM® (2016)

Název pole	Popis
ID	Identifikátor pacienta
Clump	Shluková tloušťka (clump thickness)
UnifSize	Jednotnost buněčné velikosti (uniformity of cell size)
UnifShape	Jednotnost buněčného tvaru (uniformity of cell shape)
MargAdh	Marginální přilnavost (marginal adhesion)
SingEpiSize	Velikost jedné epitelové buňky (single epithelial cell size)
BareNuc	Nahé jádro (bare nuclei)
BlandChrom	Jemný chromatin (bland chromatin)
NormNucl	Normální jadérko (normal nucleoli)
Mit	Mitóza (mitoses)
Class	Benigní nebo maligní (benign or malignant)

Údaje jsou v textovém formátu, kde jeden řádek představuje jeden případ (pacienta), jehož naměřené hodnoty proměnných jsou odděleny čárkami. Níže je náhled souboru *cell_samples.data*:

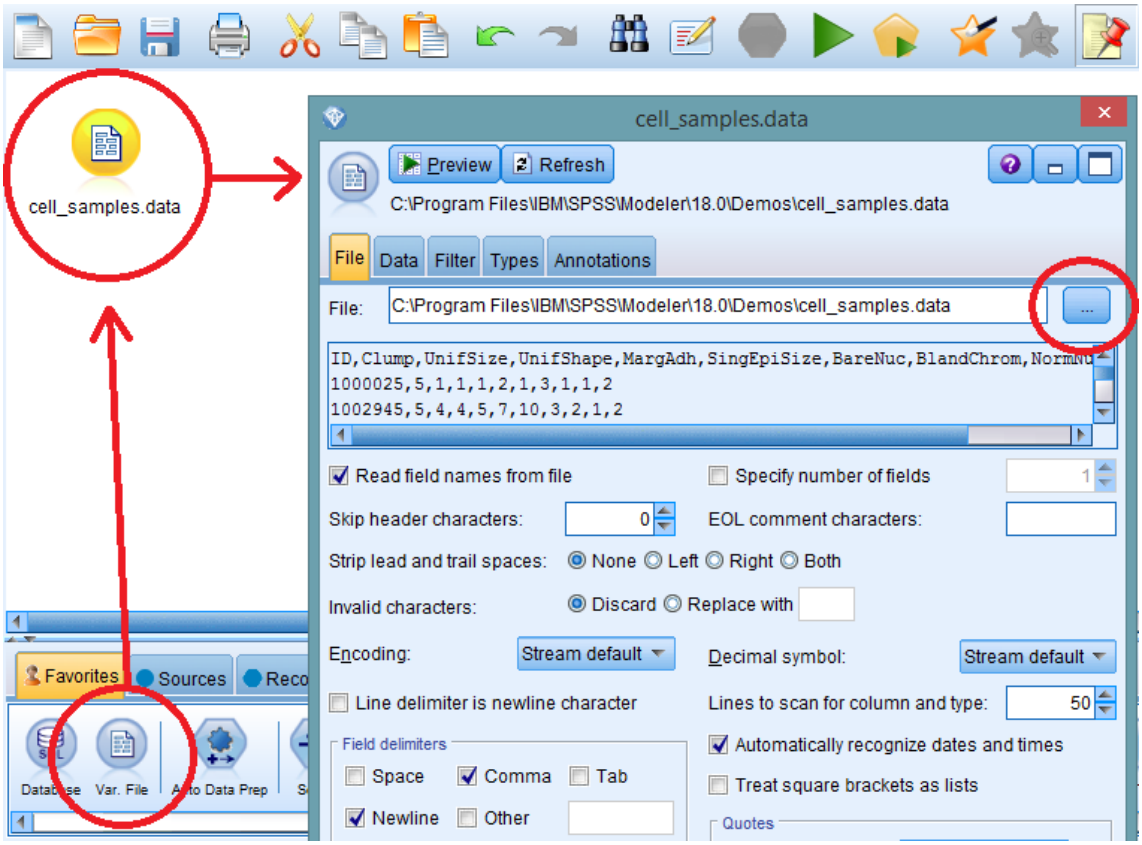
```
ID,Clump,UnifSize,UnifShape,MargAdh,SingEpiSize,...,Class
1000025,5,1,1,1,2,1,3,1,1,2
1002945,5,4,4,5,7,10,3,2,1,2
1015425,3,1,1,1,2,2,3,1,1,2
1016277,6,8,8,1,3,4,3,7,1,2
1017023,4,1,1,3,2,1,3,1,1,2
1017122,8,10,10,8,7,10,9,7,1,4
1018099,1,1,1,1,2,10,3,1,1,2
1018561,2,1,2,1,2,1,3,1,1,2
1033078,2,1,1,1,2,1,1,1,5,2
1033078,4,2,1,1,2,1,2,1,1,2
...
```

Následuje samotná práce s IBM SPSS Modeler. Cílová podoba streamu je zachycena na následujícím obrázku:



Obrázek 6 Cílová podoba streamu

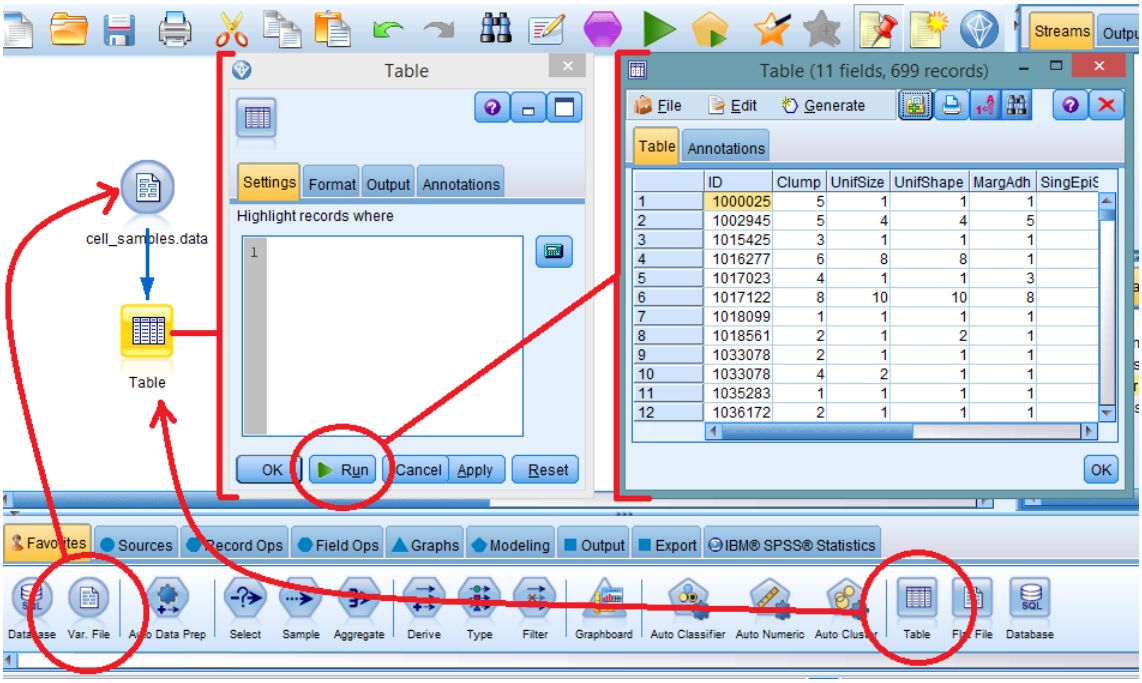
Po založení nového projektu (streamu) je třeba vytvořit uzel typu *Var. File*, do něhož se následně načte soubor s daty (*cell_samples.data*).



Obrázek 7 Načtení dat v uzlu Var. File

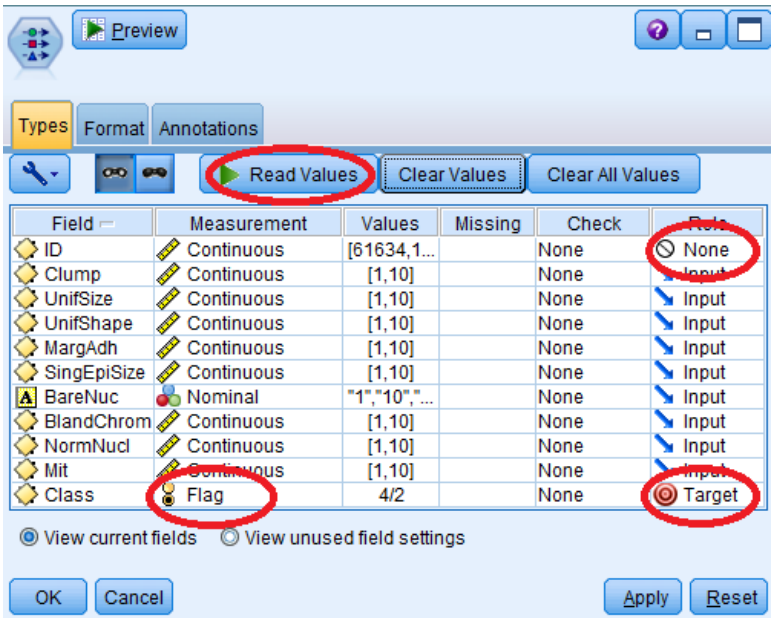
Po přidání dalšího uzlu *Table* a jeho propojením s *Var. File* je možno spustit stream a prohlédnout si načtená data v tabulce. Tabulky se dají přidávat k různým

uzlům, kde má smysl zobrazovat takto formátovaná data. Stejně tak může být použita široká škála grafů.



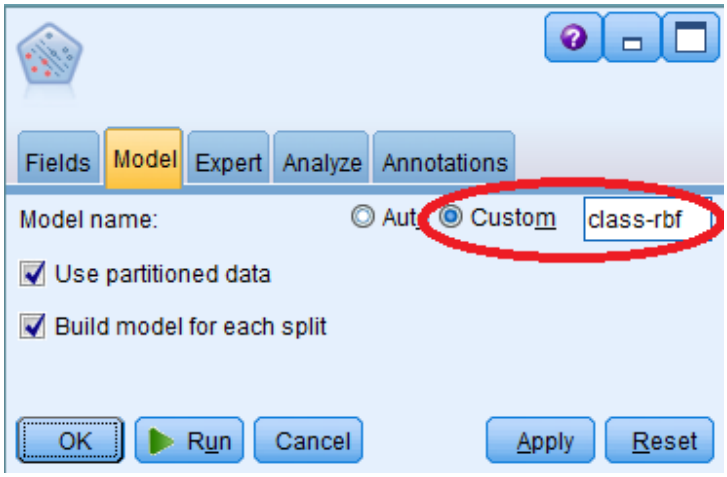
Obrázek 8 Zobrazení načtených dat v tabulce

Po vytvoření dalšího uzlu *Type* a jeho konexi s datovým uzlem, musí být vykonáno určité nastavení *Type* uzlu, aby se dosáhlo požadovaného chování. Ve sloupci *Measurement* nastavit hodnotu pole *Class* na *Flag* (*Class* má jen dvě možné hodnoty), načíst hodnoty tlačítkem *Read Values*, nastavit roli *ID* na *none* (nemá úlohu prediktoru ani cíle) a nastavit roli *Class* na *Target* (cílové pole pro diagnózu).



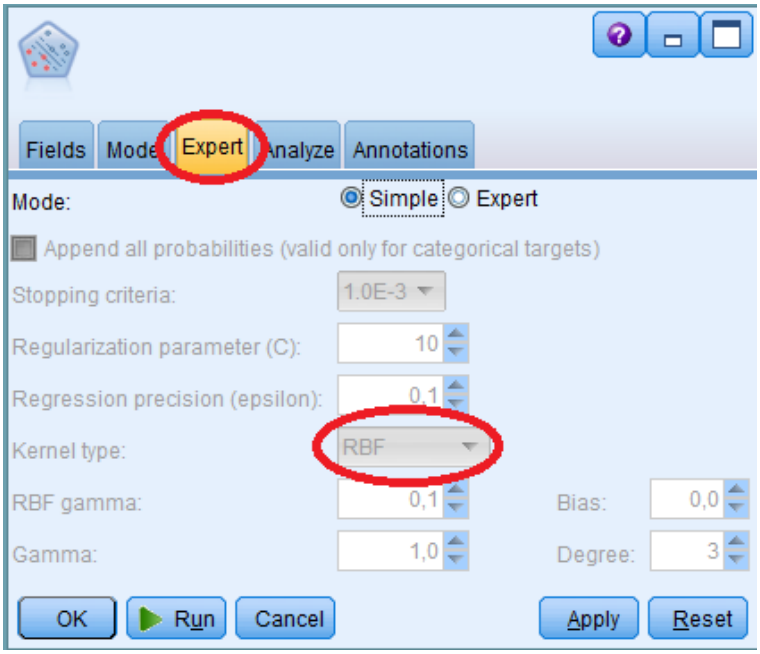
Obrázek 9 Nastavení uzlu Type

Z palety se vloží další uzel *SVM* a propojí se s uzlem *Type*. *SVM* nabízí volbu funkce kernelu pro zpracování. Protože se nedá jednoduše zjistit, která funkce je nejvhodnější pro jakoukoliv datovou sadu, je možné použít více funkcí a porovnat je. První funkcí bude *RBF* (Radial Basis Function). Na kartě *Model* v nastavení *SVM* uzlu, je vhodné pojmenovat ho podle zvolené funkce: *Model name* nastavit na *Custom* a zapsat například *class-rbf*.



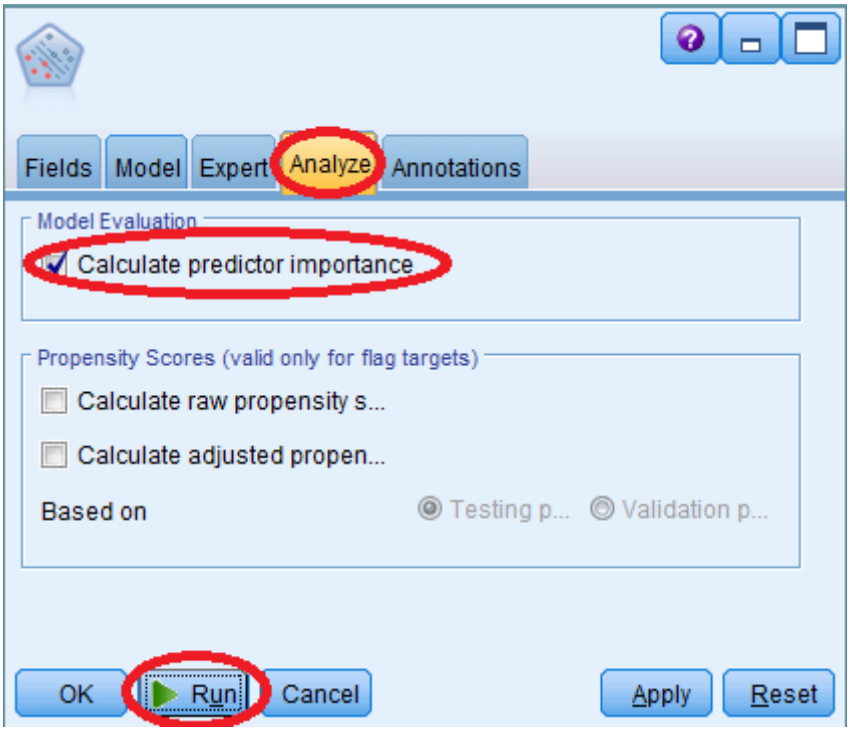
Obrázek 10 Nastavení uzlu SVM (Model)

Na kartě *Expert* by mělo být ve výchozím nastavení vše v pořádku. *Kernel type* je nastavený na *RBF.*

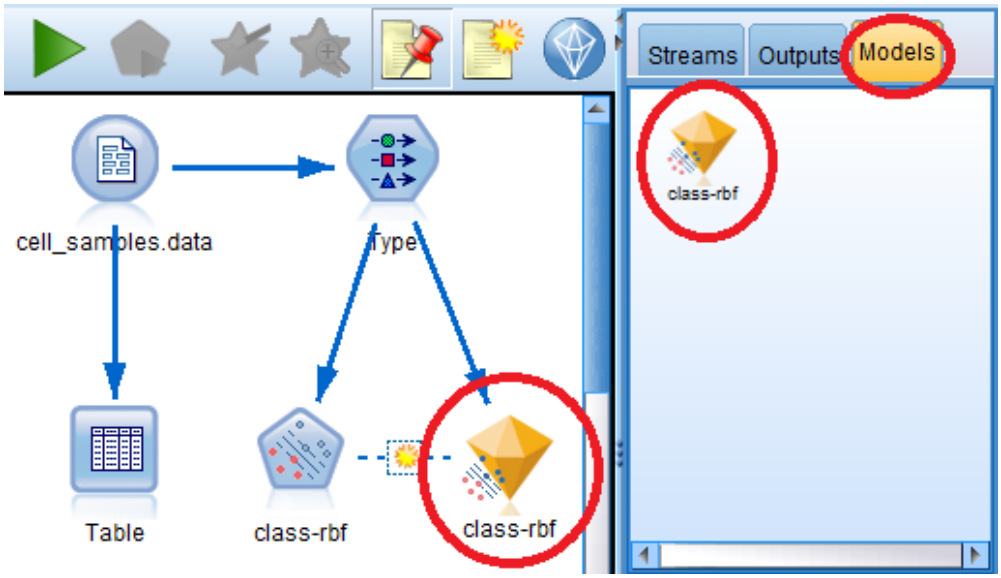


Obrázek 11 Nastavení uzlu SVM (Expert)

V záložce *Analyze* je třeba zaškrtnout pole *Calculate predictor importance* (vypočítat váhu prediktorů) a stisknout tlačítko *Run*, čímž se na plátně vytvoří další uzel označovaný jako *model nugget* (ikona vypadá jako žlutý drahokam), který je také přístupný ze správce modelů.

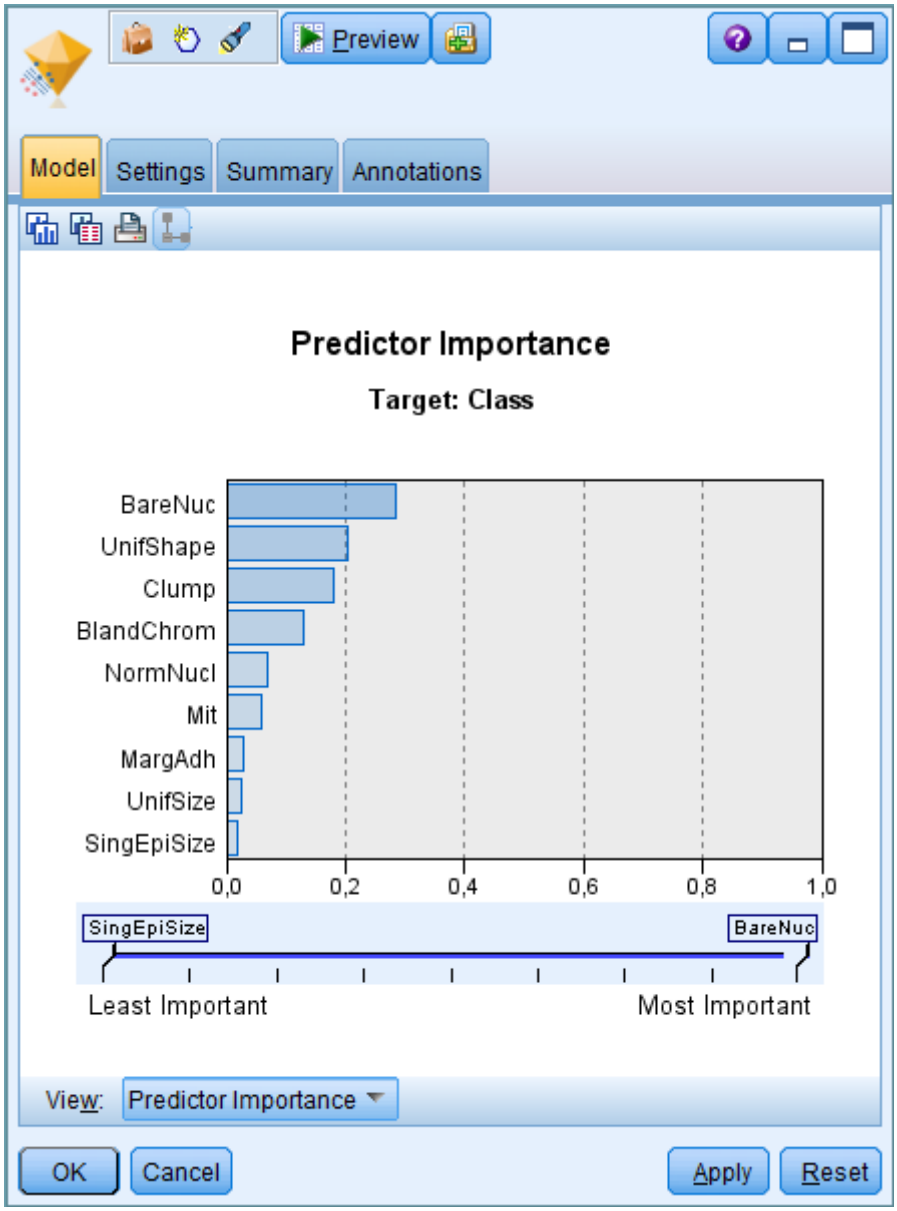


Obrázek 12 Nastavení uzlu SVM (Analyze)



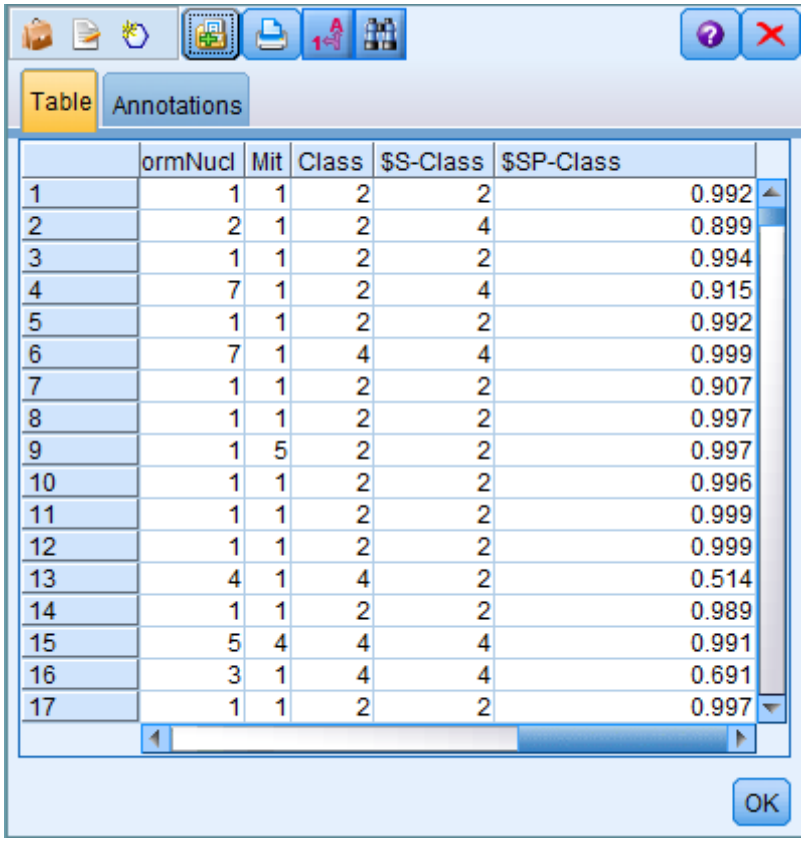
Obrázek 13 Vytvořený model nugget

Po otevření tohoto nuggetu můžeme vidět graf, který znázorňuje relativní efekt jednotlivých proměnných. Ukazuje, že největší váhu má proměnná *BareNuc*, *BlandChrom* je přibližně ve středu a trio *MargAdh, UnifSize* a *SingEpiSize* má velmi malý efekt.



Obrázek 14 Porovnání váhy proměnných

Následuje přidání dalšího *Table* uzlu a jeho propojení s *class-rbf* nuggetem. Po otevření nové tabulky je možné vidět dvě nově vytvořená pole: *$S-Class* a *$SP-Class. $S-Class* ukazuje předpovězené hodnoty proměnné *Class* a *$SP-Class* potom samotnou pravděpodobnost správnosti této predikce (od 0 do 1).

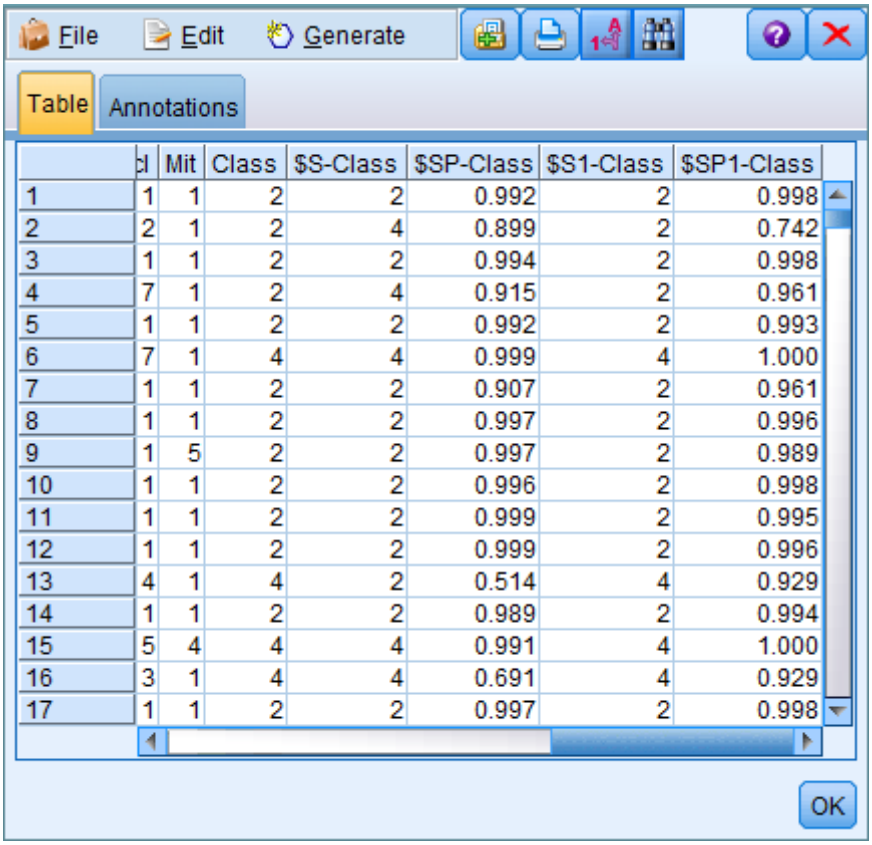


	ormNucl	Mit	Class	$S-Class	$SP-Class
1	1	1	2	2	0.992
2	2	1	2	4	0.899
3	1	1	2	2	0.994
4	7	1	2	4	0.915
5	1	1	2	2	0.992
6	7	1	4	4	0.999
7	1	1	2	2	0.907
8	1	1	2	2	0.997
9	1	5	2	2	0.997
10	1	1	2	2	0.996
11	1	1	2	2	0.999
12	1	1	2	2	0.999
13	4	1	4	2	0.514
14	1	1	2	2	0.989
15	5	4	4	4	0.991
16	3	1	4	4	0.691
17	1	1	2	2	0.997

Obrázek 15 Tabulka s výsledky funkce RBF

Zkoumáním tabulky je možno zjistit, že pro většinu záznamů je pravděpodobnost poměrně vysoká, ale jsou zde i určité výjimky, například v řádku 13 je vidět pravděpodobnost 0.514, což není nejlepší výsledek. Navíc, porovnáním sloupců *Class* a *$S-Class,* je vidět dost chybných určení (na obrázku řádky 2, 4 a 13 a další nezobrazené).

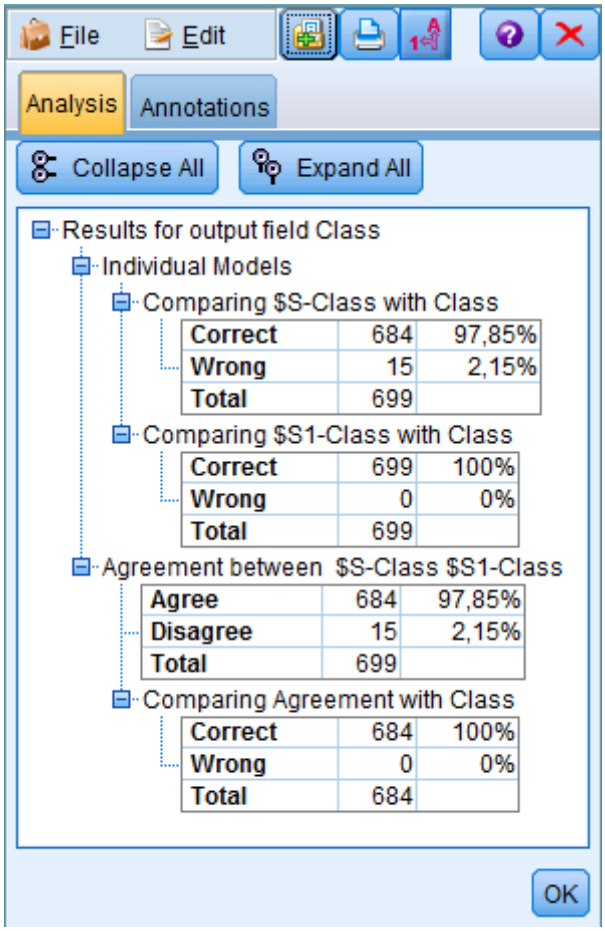
Jak už bylo řečeno, bude vytvořen další model, tentokrát s funkcí *Polynomial*. Postup se téměř shoduje s předchozím. Vytvořit další *SVM* uzel a připojit ho k *Type* uzlu. V nastavení uzlu *SVM* ho pojmenovat *class-poly*, na záložce *Expert* vybrat přepínač *Expert* a nastavit typ kernelu na *Polynomial*. Na záložce *Analyze* již nic nezaškrtávat a rovnou spustit *Run*. Vytvoří se nový model nugget *class-poly*. Tento nugget se propojí s předchozím nuggetem *class-rbf* a ve varovném okně se vybere možnost *Replace* pro nahrazení konexe. K novému nuggetu se připojí tabulka a spustí se.



	cl	Mit	Class	$S-Class	$SP-Class	$S1-Class	$SP1-Class
1	1	1	2	2	0.992	2	0.998
2	2	1	2	4	0.899	2	0.742
3	1	1	2	2	0.994	2	0.998
4	7	1	2	4	0.915	2	0.961
5	1	1	2	2	0.992	2	0.993
6	7	1	4	4	0.999	4	1.000
7	1	1	2	2	0.907	2	0.961
8	1	1	2	2	0.997	2	0.996
9	1	5	2	2	0.997	2	0.989
10	1	1	2	2	0.996	2	0.998
11	1	1	2	2	0.999	2	0.995
12	1	1	2	2	0.999	2	0.996
13	4	1	4	2	0.514	4	0.929
14	1	1	2	2	0.989	2	0.994
15	5	4	4	4	0.991	4	1.000
16	3	1	4	4	0.691	4	0.929
17	1	1	2	2	0.997	2	0.998

Obrázek 16 Tabulka s výsledky funkce Polynomial

Tabulka se rozrostla o další dvě pole: *$S1-Class* a *$SP1-Class.* V porovnání s předchozími výsledky, vykazuje funkce *Polynomial* lepší výsledky. Pro potvrzení tohoto názoru se hodí přidat další uzel typu *Analysis* a spojit ho s nuggetem *class-poly*. Následně uzel *Analysis* otevřít a kliknout na *Run.*



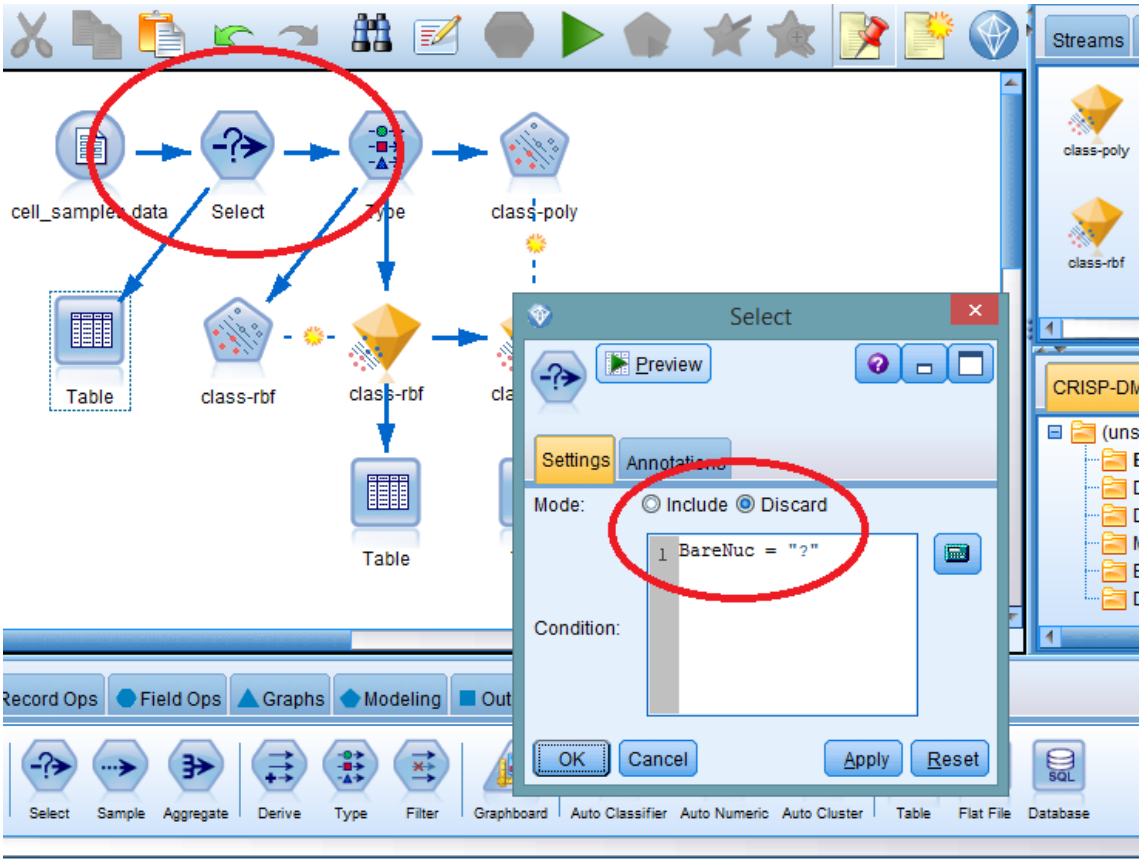
Obrázek 17 Srovnání funkcí v uzlu Analysis

Zobrazené okno dovoluje porovnávat dva a více nuggetů stejného typu. Výstup ukazuje, že funkce *RBF* správně určila diagnózu pro 97,85 % případů, zatímco funkce *Polynomial* má úspěšnost 100 %. V tomto případě by si ani ostatní dostupné funkce (*Sigmoid* a *Linear*) nevedli tak dobře, což nemusí platit pro jiné datové sady, a proto je vhodné vyzkoušet všechny možnosti. Ke stejným výsledkům lze dojít i jinými postupy modelování. Vytvořené modely se dají kdykoliv použít znovu pro jiná data.

V tomto příkladu se dokumentace nezabývala chybějícími hodnotami. Data však obsahují šestnáct případů, kdy je proměnná *BareNuc* nastavená na hodnotu „?“. Proto je také na obrázku 9 vidět, že je typ tohoto pole automaticky nastaven na nominální a systém tedy pracoval s otazníkem jako s další hodnotou. Existuje řada řešení. Jelikož je chybějících hodnot „jen“ šestnáct (asi 2,29 %) bude zde předveden jednoduchý postup pro odstranění celých záznamů s chybějícími

hodnotami a analýza bude probíhat jen s kompletními daty (tento postup již není obsažen v dokumentaci).

Stačí vložit mezi uzly pro načtení dat (*Var. File*) a *Type* další uzel *Select*. V jeho nastavení přepnout *Mode* na *Discard* a zapsat podmínku *BareNuc = "?"*. Uzel pak nepropustí záznamy obsahující otazník a analýza pokračuje bez nich.



Obrázek 18 Výběr dat podle podmínky v uzlu Select

Vymazání nekompletních záznamů nepřineslo příliš velké změny, ale je zajímavé, že funkce *RBF* v tomto případě určila hodnoty *Class* špatně v šestnácti případech, zatímco s předchozími daty, kde hrál roli hodnoty i otazník, jen v patnácti případech. Celkově tak klesla její úspěšnost z 97,85 % na 97,66 %. Zbytek zůstává stejný (kromě celkového množství záznamů, kterých nyní bylo 683).

Software IBM SPSS Modeler byl pro tuto práci vybrán z důvodu snadné ovladatelnosti a přehlednosti pro ukázku problematiky datového modelování.

8 Principy praktických řešení pro rozsáhlá data

Při navrhování a realizaci platformy pro big data je důležité držet se určitých vlastností, které by systém měl splňovat. Kvůli komplexitě řešení je důležitá robustnost, odolnost proti chybám a minimální nároky na údržbu. Rozsáhlé množství dat vyžaduje škálovatelnost systému. Aby byl systém schopen zpracovat různorodá data, měl by být v tomto ohledu navržen obecně a s možností rozšíření o další funkcionalitu. V této kapitole jsou popsány požadované vlastnosti a Lambda architektura, která je schopna je splnit a proč. Dále je stručně popsán jeden vybraný systém pro každou vrstvu, který se pro ni hodí a nakonec je uveden příklad, který naznačuje postupy pro větší efektivitu zpracování dat. Kapitola je zpracována podle Marze a Warrena (2015).

8.1 Vlastnosti big data systému

Robustnost a odolnost proti selhání

Navržený distribuovaný systém by měl být co nejodolnější proti selhání softwarových, hardwarových nebo lidských součástí. Dá se očekávat, že při desítkách či stovkách spolupracujících počítačů mohou některé z nich přestat pracovat a řešení s tímto musí počítat. Odolnost systému proti lidským chybám je mnohdy opomíjena.

(Marz a Warren, 2015)

Nízká latence při čtení a aktualizacích

Různé aplikace a procesy se liší ve svých časových potřebách pro čtení dat, jejich aktualizaci či zápis. Podle fungování a požadavků organizace je třeba zajistit vhodné odezvy systémů. V některých případech může jít o milisekundy, v jiných zase o hodiny či dny.

(Marz a Warren, 2015)

Škálovatelnost

Protože množství dat stále přibývá, řešení systému musí počítat i s rozšířením o další výpočetní zdroje. Dalšími výpočetními zdroji se zde myslí

přidání dalších počítačů (horizontální škálování) na rozdíl od vertikálního škálování, kdy se stávající počítače vymění nebo upgradují.

(Marz a Warren, 2015)

Generalizace

Čím méně bude systém specializovaný, tím nabídne flexibilnější podporu pro různé aplikace (Marz a Warren, 2015).

Rozšiřitelnost

Přidat do systému novou funkcionalitu může být někdy složitý a drahý proces, pokud na to stávající řešení není připraveno. S tímto často souvisí migrování dat do nového formátu, které by mělo být rychlé a snadné.

(Marz a Warren, 2015)

Ad hoc dotazování

Možnost ad hoc dotazování nad daty je velmi důležitá. Velké datové sady mohou ukrývat cenné informace, které by zůstaly skryty, pokud by systém neumožňoval se na ně zeptat.

(Marz a Warren, 2015)

Minimální údržba

Čím víc je systém komplexní, tím víc se zvyšuje možnost různých selhání a potřeba údržby pro jeho bezproblémový chod. Pro udržení správného fungování je třeba také vědět, kdy systém rozšířit. Aby byla potřeba údržby co nejmenší, komponenty systému by měly být, v rámci možností, co nejjednodušší. Lambda architektura se snaží přenést komplexnost do takových částí systému, jejichž výstupy jsou během pár hodin nepotřebné a zachovat jednoduchost v jádru systému.

(Marz a Warren, 2015)

Možnosti ladění

Každý systém by měl nabízet možnosti a informace, potřebné pro řešení problémů nebo ladění. Ze záznamů procesů systému musí být možnost vystopovat důvod každé hodnoty a chyby.

(Marz a Warren, 2015)

8.2 Lambda architektura

Pro splnění předešlých nároků na systém je vhodné použít architekturu, která je označována jako Lambda Architecture. Tato architektura je rozdělena do tří vrstev: speed layer, serving layer a batch layer. Dohromady tyto vrstvy zajistí prostředí pro splnění všech požadovaných vlastností sytému.

8.2.1 Vrstvy architektury

Batch layer

Slouží pro předzpracování dat, takže urychluje odpovědi na dotazy a šetří výpočetní zdroje. Pokud je například potřeba zjistit počet návštěv určité URL adresy v určitém časovém rozmezí, nemusí systém procházet znovu všechna data a hledat jednotlivé návštěvy, díky předzpracování a indexaci už se pracuje pouze s pohledy (batch view). Pro dávkové předzpracování dat se hodí například systém Hadoop, který je podrobněji popsán v další kapitole. Dávkové předzpracování je psáno ve formě jednovláknových programů, což usnadňuje jejich paralelizaci a škálování.

(Marz a Warren, 2015)

Serving layer

Po tom, co jsou připraveny jednotlivé pohledy vrstvou batch layer, je třeba je zpřístupnit uživateli. Tímto se zabývá serving layer, což je v podstatě distribuovaná databáze, do níž se nahrají pohledy a může se k nim náhodně přistupovat ve smyslu čtení nebo aktualizací záznamů. Databáze neslouží pro náhodné zápisy dat. Díky tomu je robustní, předvídatelná, jednoduchá na nastavení a obsluhu. Příklad takovéto databáze je ElephantDB. Když jsou dostupné nové verze pohledů z vrstvy batch layer, vrstva serving layer je automaticky aktualizuje.

(Marz a Warren, 2015)

Speed layer

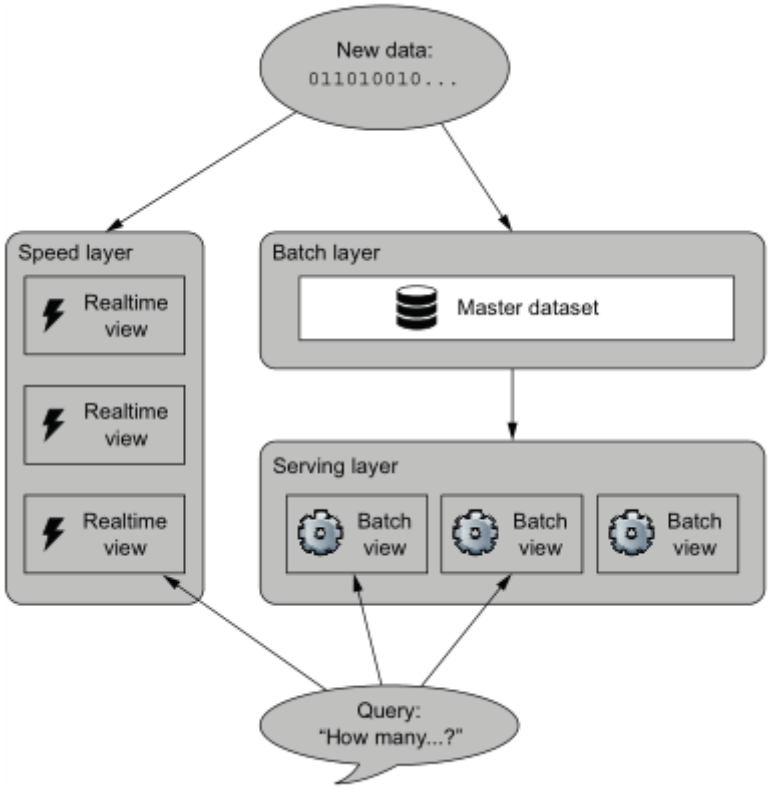
Mezitím co vrstva batch layer předzpracovává data do pohledů a vrstva serving layer je přebírá, mohou vznikat v systému nová data, která by v tuto chvíli nebyla přístupná pro dotazování. Tato data jsou zahrnována do výsledků díky speed layer vrstvě a informace jsou tedy vždy aktuální. Speed layer funguje podobně jako batch layer, ale pracuje s aktuálními daty. Aby byla zajištěna co nejvyšší rychlost, probíhá zpracování aktuálních pohledů inkrementálně, na rozdíl od batch layer, která zpracovává (zpravidla) znovu všechna dostupná data.

(Marz a Warren, 2015)

8.2.2 Širší pohled na architekturu

Jak je vidět na obrázku níže, nová data proudí do vrstev speed layer i batch layer. Batch layer předzpracuje z hlavní databáze jednotlivé pohledy, které jsou následně indexovány a převzaty do serving layer. Speed layer zpracovává pohledy pro aktuální data. Při dotazování jsou pak pohledy kombinovány.

(Marz a Warren, 2015)



Obrázek 19 Lambda architektura
Zdroj: Marz a Warren (2015)

Při využití Hadoop, jako součásti architektury, je zajištěna **robustnost a odolnost** proti selhání, protože se dokáže vyrovnat s hardwarovými i softwarovými poruchami a při lidském selhání je možné upravit algoritmus nebo se zbavit chybných dat a znovu je vytvořit. Vrstvy batch layer a serving layer jsou distribuované systémy a dovolují jednoduchou **škálovatelnost**. Lambda architektura je sama o sobě obecná, a proto umožňuje zpracovávat libovolné pohledy z libovolných dat (splňuje **generalizaci**). Master dataset (hlavní databáze) je schopna pojmout jakýkoliv typ dat. Pohledy mohou být upravovány nebo vytvářeny nové, stejně jako funkce. Tím je zajištěna **rozšiřitelnost**. Batch layer je stavěna pro **ad hoc dotazování**, všechna data jsou také dostupná z hlavní databáze. Hadoop, který je, v tomto případě, hlavní komponentou, není náročný na údržbu, pokud má správce příslušné administrativní znalosti. Serving layer není složitá, protože neumožňuje náhodný zápis a má méně součástí, kde by mohlo dojít k chybám. Tím jsou sníženy nároky na **údržbu** systému. V navrženém řešení jsou vždy k dispozici vstupní data z hlavní databáze i výstupní data v podobě pohledů.

To platí i pro případ mezikroků. **Možnosti ladění** jsou proto dostupné a jednodušší. Vstupní i výstupní data nejsou většinou dostupná v tradičních databázích, kde aktuální hodnota nahrazuje minulou. **Nízkou latenci** zajišťuje speed layer vrstva, která obsahuje databáze s možností rychlého přístupu k datům včetně náhodného zápisu. Takové systémy jsou komplexnější a náchylnější k chybám. Proto je vlastností Lambda architektury izolace komplexity, což znamená, že komplexita je přenesena na tu část systému, jejíž výstupy jsou potřebné jen dočasně. Poté, co jsou v batch layer pohledy s daty z aktuálních databází zpracované, nejsou ve speed layer už aktuální pohledy potřebné a lze se jich zbavit.

(Marz a Warren, 2015)

8.2.3 Příklad systémů pro jednotlivé vrstvy

Hadoop

Apache Hadoop, jehož hlavními součástmi jsou distribuovaný souborový systém HDFS a programovací model MapReduce pro distrubuované zpracování dat, se pro Lambda architekturu velmi hodí. HDFS hraje roli hlavní databáze, která je vhodná pro zápis a čtení rozsáhlých datových souborů, dovoluje paralelní zpracování dat, udržuje jejich repliky, takže riziko ztráty dat je minimální a je jednoduše škálovatelná. Stejně tak snadná škálovatelnost MapReduce dovoluje psát programy pro zpracování dat, které když dokážou zpracovat 10 gigabajtů, stejně dobře si poradí i s 10 petabajty dat. MapReduce automaticky paralelizuje běžící úlohy napříč výpočetními zdroji v clusteru, stará se o synchronizaci, přenos dat, plánování úloh a je odolný proti chybám. V Lambda architektuře pracuje ve vrstvě batch layer spolu s HDFS, ze kterého čerpá data a připravuje z nich jednotlivé pohledy.

(Marz a Warren, 2015)

ElephantDB

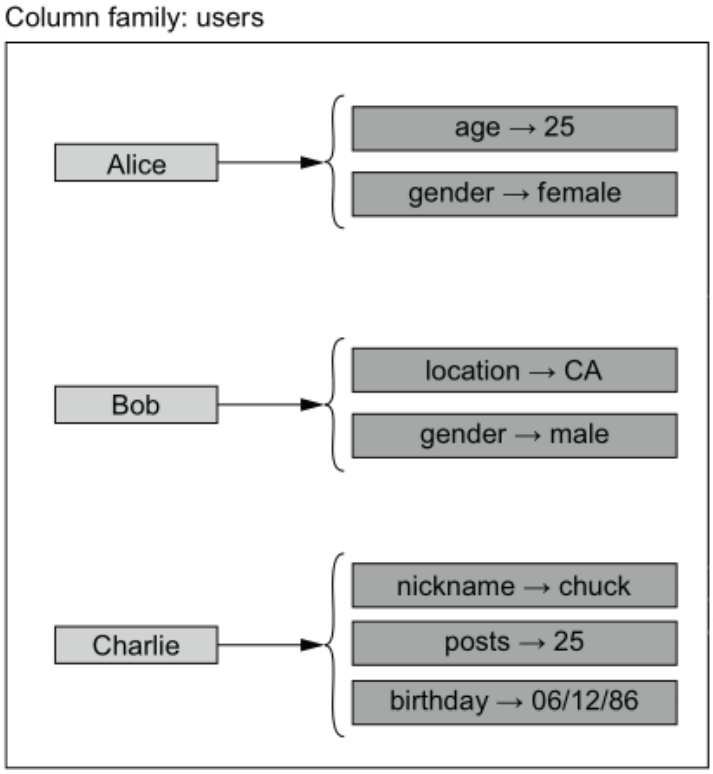
ElephantDB (EDB) pracuje ve vrstvě serving layer. Jedná se o distribuovanou databázi typu klíč/hodnota, kde obojí ukládá jako pole bajtů. Každý pohled, vycházející z batch layer, je rozdělen do fixního počtu menších částí

(shards), který je dán počtem reduce úloh na jeden pohled. Každá část je pak indexována a uložena do distribuovaného souborového systému. Jednotlivé EDB servery přebírají přidělené části pohledu (včetně jejich replik), které jsou rovnoměrně rozděleny. Servery jsou také zodpovědné za mazání starých verzí pohledů.

(Marz a Warren, 2015)

Cassandra

Pro speed layer je použita databáze Cassandra. Ta má za úkol shromažďovat aktuální data a připravovat z nich pohledy v reálném čase (Marz a Warren, 2015). Její model pro ukládání dat je zachycen na následujícím obrázku:



Obrázek 20 Datový model databáze Cassandra
Zdroj: Marz a Warren (2015), upraveno

Cassandra bývá označována jako sloupcově orientovaná databáze, ale její model ukládání je vhodnější popsat spíše jako mapu, která odkazuje na hodnoty v podobě další mapy. Jak je vidět na předcházejícím obrázku, druh sloupců (column family) je označení pro nezávislou sadu dat, podobně jako tabulka

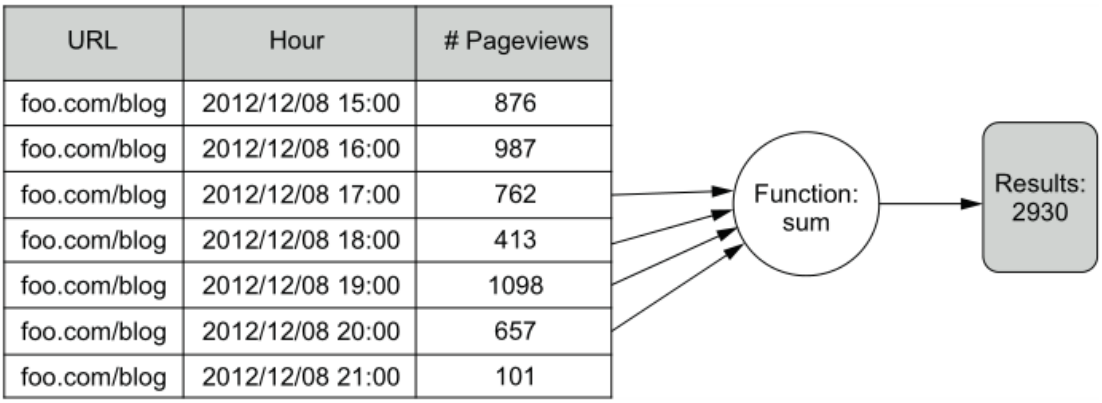
v relačních databázích. V tomto případě obsahuje informace o uživatelích (users) ve formě klíč-hodnota. Tyto hodnoty tvoří pár název-hodnota a jsou označovány jako sloupce (columns). Jsou uloženy fyzicky na jednom místě (což urychluje přístup) a jejich množství a obsah se mohou lišit klíč od klíče. Jeden klíč může odkazovat na tisíce, nebo i miliony sloupců.

(Marz a Warren, 2015)

8.2.4 Příklad: počet přístupů na URL adresu

Tento příklad demonstruje, jaké postupy mohou být vykonány pro efektivnější práci systému při dotazování se na počet návštěv určité URL adresy.

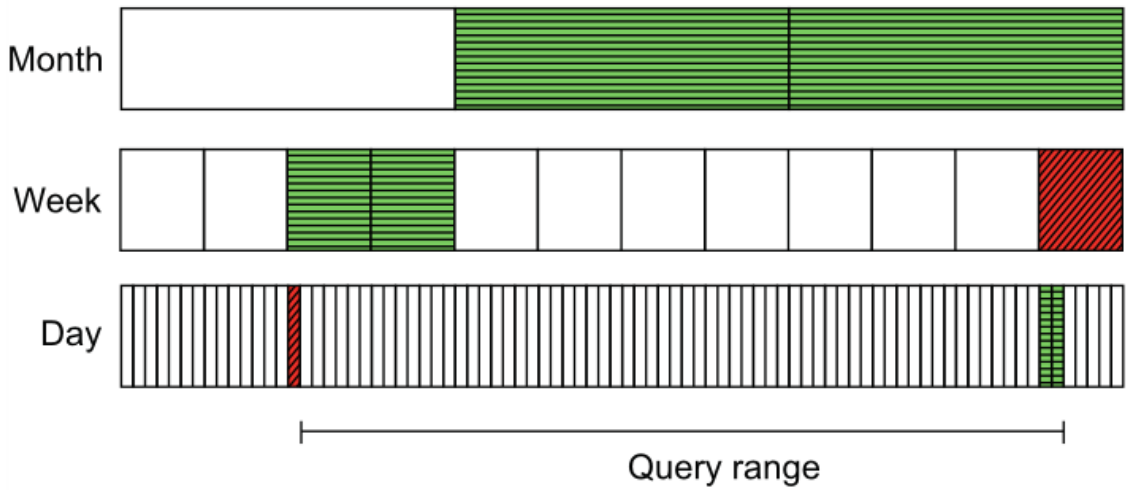
Uživatel se dotazuje na počet přístupů na web v určitém časovém rámci s rozlišením na úrovni jednotek hodin. Nejjednodušším řešením je připravit pohledy pro každou URL adresu za každou hodinu. Takto zpracovaný pohled může vypadat následovně:



Obrázek 21 Zpracovaný pohled na úrovni hodin
Zdroj: Marz a Warren (2015)

Při dotazování jsou pak sečteny počty návštěv v požadovaném intervalu. V případě většího rozsahu se však může významně navýšit čas potřebný pro zpracování. Pro tyto případy se vytvářejí pohledy na různých stupních granularity, například úroveň dnů, týdnů nebo měsíců, které se při dotazování kombinují. Princip je znázorněn na obrázku:

(Marz a Warren, 2015)



Obrázek 22 Kombinace různých stupňů granularity
Zdroj: Marz a Warren (2015)

Časový interval je například od 3. března 3:00 do 17. září 8:00. Na úrovni hodin by bylo třeba sčítat 4 805 hodnot, ale s pohledy různých granularit jen 26. Jak ilustruje obrázek výše, při dotazu se vezmou hodnoty za jednotlivé měsíce, ke kterým jsou přičteny nebo naopak odečteny (červená barva na obrázku) hodnoty z jednotlivých týdnů, dnů a hodin, aby bylo dosaženo sumy z požadovaného rozmezí. Výhoda není jen v ušetření výpočetních sil počítačů, ale také se snižuje množství dat, potřebných k přenášení. To je důležité, protože se jedná o distribuovaný systém, kde jsou data uložena na různých strojích, discích a jejich blocích. Potřebný čas pro přenesení dat ovlivňují různé faktory, například vytíženost sítě nebo serveru, přístupová doba a rychlost disku a podobně. Proces pak stojí na nejpomalejším zapojeném článku. Alternativním nebo doplňujícím řešením může být využití indexace a řazení, kdy jsou data uložena fyzicky na jednom místě. Dalšího zvýšení výkonu je možné dosáhnout například použitím SSD disků, které jsou však podstatně dražší v porovnání s klasickými plotnovými disky.

(Marz a Warren, 2015)

9 Apache Hadoop

O Hadoop se mluví několik let a diskuse přešla od akademických úvah do řešení reálných problémů. Z technického hlediska je to díky „vyzrálosti“ systému a také vzniku jeho distribucí. Systém se totiž, kromě klíčových modulů, může skládat z mnoha dalších částí a jeho poskládání může být složité (například kvůli nekompatibilitě různých verzí komponent, které se rychle vyvíjejí). Hadoop distribuce jsou již předpřipraveným systémem, který je možné využít jako platformu pro zpracování big data. Hadoop je v základu určen především pro práci s velkým množstvím nestrukturovaných dat. Pro splnění požadavků big data je kombinován i s tradičními systémy, například relačními databázemi, které mají řadu opačných vlastností a hodí se zase pro jiné úlohy.

(Augustin, 2014)

Na otázku, proč právě teď začít používat Hadoop, odpovídá citace:

„Nízká úvodní investice i provozní náklady, vstřícná licenční politika, případy užití slibující zvýšení konkurenceschopnosti, to jsou hlavní důvody, proč právě teď začít s Hadoop ekosystémem. Dalším driverem bude snaha o vybudování vlastního know-how a zaškolení interních zdrojů, protože počet zkušených a dostupných ‚big data‘ specialistů na českém trhu je nízký.“

(Augustin, 2014)

Apache Hadoop má své kořeny v projektu Apache Nutch, což je webový vyhledávač, tvořící část projektu Apache Lucene – široce používanou knihovnu pro textové vyhledávání. Architektura Apache Nutch však postrádala dostatečnou škálovatelnost pro webové měřítko, a proto vznikla open source implementace souborového systému od Googlu (Google Distributed Filesystem – GFS), pojmenovaná Nutch Distributed Filesystem (NDFS). Podobně Google později představil framework MapReduce pro distribuované zpracování dat, který byl vývojáři Nutch implementován a použit, podobně jako NDFS. Díky tomu dosáhly možnosti Apache Nutch dál, než by se dalo využít v oblasti vyhledávání, a proto vznikl projekt Apache Hadoop.

(White, 2012)

Společnost Apache Software Foundation, pod kterou Hadoop spadá, udává popis na oficiálních stránkách Welcome to Apache™ Hadoop®! (2014), podle kterého je cílem projektu Apache Hadoop vývoj open source softwaru pro spolehlivé, škálovatelné a distribuované výpočty. Píše o Hadoop jako o softwarové knihovně a frameworku, který umožňuje distribuované zpracování velkých datových setů napříč počítačovými clustery (skupina počítačů spolupracujících prostřednictvím počítačové sítě) pomocí jednoduchých programovacích modelů. Může běžet jak na jednom, tak na tisících počítačích, které nabízejí lokální výpočetní a úložné zdroje. Není třeba využívat špičkový (high-end) hardware, ale běžně dostupný komoditní hardware, který je sice náchylnější k poruchám, ale vysoká dostupnost je zajištěna na aplikační vrstvě. Projekt Hadoop zahrnuje tyto hlavní moduly:

- **Hadoop Common** - základní nástroje pro podporu dalších modulů.
- **Hadoop Distributed Filesystem** (HDFS, přejmenovaný z NDFS) - distribuovaný souborový systém.
- **Hadoop YARN** (Yet Another Resource Negotiator) - plánovač úloh a správce výpočetních zdrojů.
- **Hadoop MapReduce** - systém pro paralelní zpracování dat, založený na YARN.

(Welcome to Apache™ Hadoop®! 2014)

Jak udává Dolejš (2012), Hadoop je využíván světovými společnostmi jako Yahoo! nebo Facebook, ale dá se využít i v menších firmách, v obou případech se ušetří náklady na správu dat.

Aby bylo zajištěno rychlé čtení a zápis velkého množství dat v co nejkratším čase, je využíváno více pevných disků, mezi které jsou data rozdělena, a přístup k nim probíhá paralelně. Pro případ selhání hardwaru je dostupnost dat zajištěna jejich replikací. MapReduce můžeme porovnat s relačními databázovými systémy (RDBMS). Zatímco RDBMS pracují se strukturovanými daty s vysokou integritou a umožňují rychlejší změny menšího množství dat v databázi díky indexaci, MapReduce je vhodný spíše pro částečně strukturovaná nebo nestrukturovaná

data a pro zpracovávání celých datových setů, jejichž obsah se příliš nemění, ale často se k němu přistupuje. Rozdíly jsou shrnuty v tabulce níže. Tyto rozdílné přístupy jsou, mimo jiné, důsledkem fungování a vývoje pevných disků. Hraje zde roli například přístupová doba (latence) nebo rychlejší zvyšování kapacity disků v poměru k rychlosti čtení.

(White, 2012)

Tabulka 4 Srovnání RDBMS a MapReduce
Zdroj: White (2012)

	Relační databáze	MapReduce
Množství dat	Gigabajty	Petabajty
Přístup	Interaktivní a dávkový	Dávkový
Změny dat	Časté čtení a zápis	Jeden zápis, časté čtení
Struktura	Statické schéma	Dynamické schéma
Integrita	Vysoká	Nízká
Škálovatelnost	Nelineární	Lineární

9.1 HDFS

Hadoop Distributed Filesystem (HDFS) je distribuovaný souborový systém, navržený pro fungování v rámci počítačového clusteru. Oproti běžným systémům souborů pro jeden disk, které mají velikost bloku disku většinou 4 - 32 kB, ukládá HDFS soubory v blocích o velikosti 64 MB (defaultně). Důvodem je preference propustnosti před latencí – je vhodný pro čtení velkých souborů. HDFS je optimalizován pro úkoly typu „zapiš jednou, čti mnohokrát“. Pro případ selhání disku jsou soubory replikovány na jiném počítači v rámci clusteru.

(Turkington, 2013)

Systém HDFS není vhodný, pokud požadujeme nízkou latenci nebo častou modifikaci souborů. Jeden soubor může teoreticky obsadit celou úložnou kapacitu clusteru, ale není možné ukládat příliš mnoho malých souborů. Důvodem je počítač označovaný jako NameNode, který ukládá metadata souborového systému do operační paměti, a záleží tedy na její kapacitě. Milion souborů o velikosti jednoho bloku zabere přibližně 300 MB kapacity paměti.

(White, 2012)

Bloky disku

Systém souborů HDFS rozděluje disk na bloky, v defaultním nastavení o velikosti 64 MB. Větší soubory se dělí do více nezávislých bloků, které nemusí být fyzicky na stejném úložišti. Díky tomu může být soubor větší než jakýkoliv disk v síti. Protože jednotku abstrakce tvoří blok fixní velikosti a ne soubor, zjednodušuje se správa ukládání a snižuje se množství metadat. Typicky je každý blok replikován na třech fyzicky oddělených počítačích pro případ havárie.

(White, 2012)

Velikost bloků lze nastavovat podle velikosti jednotlivých souborů, takže se poslední blok zmenší a neplýtvá se tak místem (Wadkar, Siddalingaiah a Venner, 2014).

NameNode a DataNode

Počítačový cluster je tvořen počítači, které představují uzly (nodes). Názvy uzlů se odvíjejí od názvu procesu, který na nich běží. Nadřízený uzel, zvaný **NameNode** (NN), spravuje jmenný prostor souborového systému HDFS. Udržuje metadata o souborech a adresářích, tvořících stromovou strukturu, a poskytuje uživateli jednotný pohled na data. Na uzlech v úloze **DataNode** (DN), které jsou podřízené nadřízenému NameNode, se nacházejí samotná data. Pokud klient potřebuje pracovat se soubory, od NN získá jen potřebná metadata a přistupuje pak k DN uzlům. Samotná data už přes NN neproudí. Při ztrátě metadat z NN uzlu jsou data na DN uzlech nepoužitelná. DN posílají NN uzlu pravidelně zprávu o své funkčnosti (heartbeat), aby se předešlo odkazování na nefunkční uzel v případě poruchy.

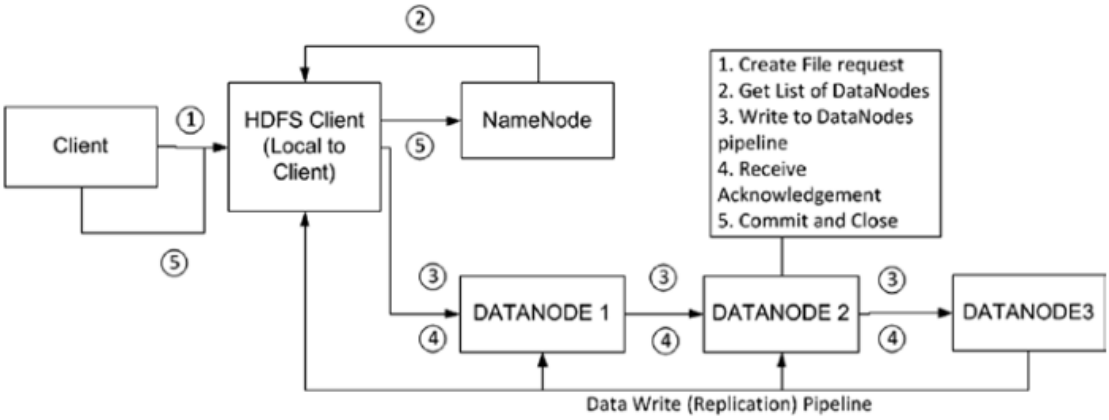
(Wadkar, Siddalingaiah a Venner, 2014)

Skriptem *sbin/start-dfs.sh* se služby NameNode a DataNode spustí. Webové rozhraní NN uzlu je na portu 50070 (*http://namenode_host:50070*). Rozhraní nabízí přehled informací o stavu HDFS, procházení souborového systému, stav DataNode uzlů, čtení logů a další.

(Lam, 2011)

Proces zápisu souboru

Na následujícím diagramu je zachycen proces zápisu. Pro tento případ je předpokládaná typická úroveň replikací dat, tedy úroveň 3.



Obrázek 23 Proces zápisu souboru v HDFS
Zdroj: Wadkar, Siddalingaiah a Venner (2014)

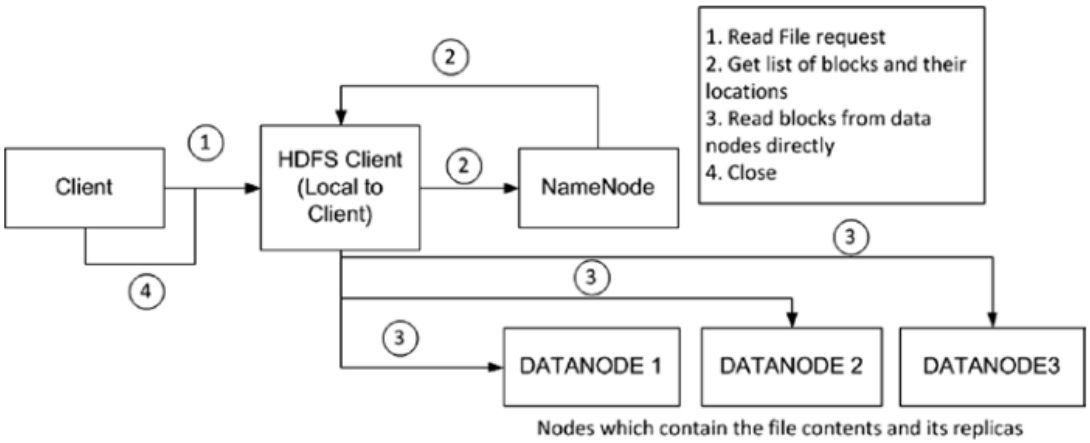
Popis jednotlivých kroků podle Wadkara, Siddalingaiaha a Vennera (2014):

1. Klient začne obsah ukládat do dočasného souboru v lokálním souborovém systému předtím, než kontaktuje NameNode.
2. Jakmile velikost dat dosáhne velikosti jednoho bloku, klient kontaktuje NameNode.
3. NameNode vytvoří soubor v HDFS a klientovi pošle identifikátory a umístění DataNode uzlů. Tato zpráva obsahuje i seznam uzlů pro replikaci dat.
4. Data jsou zapsána do DataNode uzlů a dočasný soubor na klientovi se maže.
5. NameNode potvrdí přenos a soubor se stane viditelným v systému.

(Wadkar, Siddalingaiah a Venner, 2014)

Proces čtení souboru

Podobně, jako v předchozí části, je níže uveden diagram zachycující jednotlivé fáze čtení souboru.



Obrázek 24 Proces čtení souboru v HDFS
Zdroj: Wadkar, Siddalingaiah a Venner (2014)

Popis jednotlivých kroků podle Wadkara, Siddalingaiaha a Vennera (2014):

1. Klient kontaktuje NameNode, který vrací seznam bloků a jejich lokací včetně jejich replik.
2. Klient začne číst soubor z DataNode uzlu. Pokud nastane problém, pokusí se o čtení replikovaných dat z jiného umístění.
3. Při čtení bloku se kalkuluje kontrolní součet (checksum), který se porovnává se součtem vzniklým při zápisu souboru. Pokud součty nesouhlasí, kontaktuje se jiný DataNode s replikou dat.

(Wadkar, Siddalingaiah a Venner, 2014)

Smazání souboru

HDFS může používat koš, podobně jako je známý z Windows nebo jiných systémů. Defaultně není však koš v HDFS povolen. Aktivace koše se provede přidáním vlastnosti `fs.trash.interval` do konfiguračního souboru */etc/hadoop/core-site.xml*. Jako hodnota se uvádí doba do automatického odstranění souboru z koše v minutách. Konkrétně pro nastavení koše je třeba, aby soubor *core-site.xml* obsahoval následující kód:

```
<configuration>
  <property>
    <name>fs.trash.interval</name>
    <value>600</value>
```

```
  </property>
</configuration>
```

Hodnota automatického vysypání koše (value) je nastavena na 600 minut. Podobně se nastavují mnohé další vlastnosti systému. Při smazání jsou pak soubory přesunuty do adresáře *.Trash*, který se nachází v domovské složce každého uživatele. Vymazání obsahu koše před uplynutím nastaveného intervalu se provede, v příkazové řádce, následujícím příkazem:

```
$ hdfs dfs -expunge
```

Obnovení souboru se provede jednoduše jeho přesunutím z adresáře *.Trash* na požadované místo.

(White, 2012)

Checkpoint Node a Backup Node

NameNode za běhu systému zapisuje změny v souborovém systému do log souboru *edits.* Při spouštění načte NN obraz HDFS ze souboru *fsimage* a aplikuje změny zaznamenané v souboru *edits*. Následně vznikne nový *fsimage* soubor, zahrnující změny, a dále se již pracuje s prázdným logem *edits.* Protože na NN uzlu probíhá tato operace jen při spouštění, může se po čase stát soubor *edits* příliš velkým a zapříčinit pomalý start systému.

(HDFS Users Guide, 2015)

Checkpoint Node slučuje soubory *fsimage* a *edits* pravidelně a udržuje tak velikost logu v rámci limitu. Vzniklý stav se označuje jako checkpoint. Poté je tento stav odeslán do NN uzlu. Zastaralejší forma Checkpoint Node je Secondary NameNode, který se liší tím, že nemá funkci nahrávání stavu do NN uzlu. Ten musí po checkpointu stahovat data sám. Další nevýhodou je samotný název Secondary NameNode, který je matoucí a vyvolává dojem, že se jedná o jakýsi záložní NameNode, který převezme funkci hlavního NN uzlu při jeho havárii. **Backup Node** poskytuje stejnou funkcionalitu jako Checkpoint Node, ale navíc je s NN synchronizovaný. Nemusí pravidelně stahovat *edits* soubor, protože mu ho NN

odesílá v reálném čase. Udržuje aktuální stav v operační paměti a vytváří checkpoint uložením do *fsimage* souboru.

(Hadoop: NameNode, Checkpoint Node and Backup Node, 2013)

HDFS Federation

NameNode si udržuje reference na každý soubor a blok v operační paměti, což může být problém u velkých clusterů, kde by paměť mohla být limitující faktor pro škálování. HDFS Federation umožňuje rozšíření clusteru o další NameNode uzly, kde každý spravuje část jmenného prostoru souborového systému. Jeden uzel může například spravovat soubory pod složkou */user* a druhý pod */share*. Jmenný prostor představuje adresáře, soubory a bloky. Kolekce bloků spadajících pod jeden jmenný prostor se nazývá Block Pool. Mezi NameNode uzly ve federaci neprobíhá komunikace, selhání jednoho nezapříčiní selhání druhého, pracují nezávisle na sobě.

(White, 2012)

HDFS High Availability

I při replikaci metadat na více souborových systémech a využívání Checkpoint Node, Backup Node nebo Secondary NameNode uzlu, není zajištěna vysoká dostupnost (high availability). Když nastane selhání, může u velkých clusterů vzniknout prodleva 30 i více minut při startu nového NameNode uzlu. Administrátor musí spustit nový NN, nakonfigurovat klienty a DataNode uzly pro spolupráci s novým NN uzlem, který musí znovu načíst soubory *fsimage* a *edits* do operační paměti a čekat na potvrzovací zprávy od DataNode uzlů. Hadoop od verze 2 toto řeší podporou HDFS High Availability. V této implementaci je další NN uzel v pohotovostním režimu (**Standby NameNode**) a v případě poruchy primárního NN uzlu je připraven převzít jeho funkci řádově v desítkách sekund.

(White, 2012)

Aktivní NN a Standby NameNode by měly běžet na rovnocenném hardwaru. Aby stav aktivního a pohotovostního uzlu zůstal synchronizovaný, využívá se služba zvaná JournalNode (JN). Protože je relativně nenáročná, může běžet společně s NameNode, JobTracker nebo YARN Resource Manager, nejsou pro ni

nutné samostatné počítače. Prostřednictvím JN služby se sdílí stav *edits* souboru mezi oběma NN uzly. Tyto změny musí být odeslány na většinu JN uzlů, proto je nutné, aby byly spuštěné minimálně tři oddělené instance JN. Při třech instancích je systém schopen odolat poruše jednoho počítače. Počet JN instancí může být vyšší, ale musí se jednat vždy o liché číslo, aby se zvýšila míra tolerance selhání. Pro *n* instancí je systém schopen zvládnout *(n - 1) / 2* poruch. Je také potřeba, aby Standby NameNode měl aktuální informace o DataNode uzlech, proto musí být každý DN nastaven tak, aby odesílal zprávu o funkčnosti (heartbeat) a lokacích bloků oběma NN uzlům. Při HDFS High Availability se o vytváření checkpointu stará Standby NameNode a nepoužívají se uzly Secondary NameNode, Checkpoint Node, ani Backup Node. To dovoluje využít jejich hardware pro přechod na HDFS High Availability ze systému, kde tato architektura nebyla použita.

(HDFS High Availability Using the Quorum Journal Manager, 2015)

Příkazová řádka

Jedním ze způsobů, jak komunikovat s HDFS, je příkazová řádka. Je to nejjednodušší a také velmi oblíbený způsob pro mnohé vývojáře (White, 2012). Důležité skripty pro tuto část jsou v balíku Hadoop umístěny v adresáři *bin*. Konkrétně *bin/hadoop* a *bin/hdfs*. White (2012) uvádí jako základní příkaz pro práci s HDFS `hadoop fs`, nicméně podle deRoos (2014) se jedná o zastaralý způsob a od verze Hadoop 2 se doporučuje užívat příkaz `hdfs dfs`, i když je možno užít obě varianty. K získání popisu všech příkazů pro daný skript, ho stačí pouze spustit bez dalších argumentů. Pokud jde o cestu k souboru, přijímá příkazový procesor (shell) pro souborový systém jako argumenty URI (Uniform Resource Identifier). Taková cesta je ve tvaru *scheme://authority/path*. Pokud se vynechají části *scheme* a *authority*, jsou doplněny podle nastavení v konfiguračním soboru. Část *scheme* může nabývat hodnot *hdfs* pro HDFS nebo *file* pro lokální souborový systém. Například pro cestu *hdfs://NameNode_host/parent/child* můžeme použít jen *parent/child*, pokud je *scheme* a *authority* nastaveno na *hdfs://NameNode_host*. Většina příkazů funguje podobně jako jejich ekvivalenty v unixových systémech.

(Apache Hadoop 2.7.1, 2015)

Aby bylo možno vykonávat MapReduce úlohy, je třeba vytvořit domovský adresář uživatele a složku */user*, ve které se má nacházet. Toto se provede příkazem `mkdir`:

```
$ hdfs dfs -mkdir /user
$ hdfs dfs -mkdir /user/john
```

Jako uživatelské jméno je zde použito *john*. Pro další ukázku je vytvořena složka */user/john/folder1* a z lokálního souborového sytému zkopírován již existující textový soubor *doc.txt* do adresáře *folder1* v HDFS.

```
$ hdfs dfs -mkdir /user/john/folder1
$ hdfs dfs -put doc.txt /user/john/folder1
```

Ke kopírování z lokálního systému souborů do HDFS slouží příkaz `put`. Pro opačný směr pak `get`, nebo je možné použít podobně pracující příkazy `copyFromLocal` a `copyToLocal`.

Obsah adresáře *john* vypíše příkaz `ls`.

```
$ hdfs dfs -ls /user/john/
Found 2 items
-rw-r--r--  1 john supergroup 118 2016-01-10 16:38  /user/john/doc.txt
drwxr-xr-x  - john supergroup 0 2016-01-10 16:33  /user/john/folder1
```

Jednotlivé sloupce výpisu mají následující význam:

1. První sloupec ukazuje mód souboru (file mode), „-“ pro soubor nebo „d“ pro adresář, další znaky udávají přístupová práva.
2. Druhý sloupec ukazuje nastavený faktor pro replikaci souborů (v tomto případě 1), což se nevztahuje na adresáře.
3. Ve třetím sloupci najdeme vlastníka souboru.
4. Po vlastníkovi souboru je zde uvedena skupina, do které patří.
5. Následující číslo udává velikost souboru v bajtech, u adresářů je vždy nula.
6. V šestém sloupci je zaznamenáno datum poslední změny.

7. V sedmém sloupci potom čas poslední změny.
8. Poslední údaj označuje cestu k souboru nebo adresáři.

(deRoos, 2014)

9.2 MapReduce

MapReduce (MR) je framework pro programátory, představený firmou Google v roce 2004. Umožňuje vytváření vysoce paralelizovaných a distributivních algoritmů ke zpracování velkého množství dat. Je navržen, aby fungoval na clusterech složených z běžného komoditního hardwaru. Vychází z principu „rozděl a panuj“. Jedná se o hlavní výpočetní komponentu systému Hadoop a patří mezi nejoblíbenější a nejlepší v oblasti zpracování velkého množství dat. Mnoho problémů, týkajících se například vyhledávání na bázi rozpoznávání vzorů, zpracovávání grafů nebo strojového učení, bylo díky MR vyřešeno.

(Lublinsky, Smith a Yakubovich, 2013)

Komplexitu paralelního programování skrývají pomocí abstrakce nástroje Apache Pig nebo Apache Hive (deRoos, 2014).

V porovnání s dotazovacím jazykem SQL dovoluje MR zpracovávat data ve více obecnějších podobách. Z dostupných dat se dají například vytvořit komplexní statistické modely. Pokud však potřebujeme pracovat s daty vhodnými spíše pro relační struktury, není MR příliš obratný. V tomto případě je možno využít řadu rozšíření dovolujících psát dotazy podobné SQL jazyku, které jsou pak převedeny do kódu MR.

(Lam, 2011)

MapReduce model

Model MapReduce je rozdělen do dvou kroků: Map a Reduce. V první mapovací fázi mohou být vstupy rozděleny do logických částí, z nichž může být každá zpracována nezávisle. Výstupy mohou být fyzicky rozděleny do odlišných setů, které jsou pak seřazovány. Každá seřazená část je pak předána do druhé fáze procesu (Reduce). Jedna úloha může zpracovávat paralelně jednotlivé části záznamů a zároveň může paralelně běžet více různých úloh. Pro názornost se často

udává příklad aplikace, která spočítá počet výskytů jednotlivých slov v sadě textových dokumentů.

1. Po prvním kroku Map dostaneme výstupy v podobě *<klíč, hodnota>*, kde *klíč* je slovo v dokumentu a *hodnota* je počet jeho výskytů (v tomto případě bude vždy 1, viz dále). Seznam těchto záznamů, ze zpracování jednoho dokumentu, může vypadat následovně:
    - *<"Hadoop", 1>*
    - *<"Hadoop", 1>*
    - *<"Hadoop", 1>*
    - *<"koncept", 1>*
    - *<"koncept", 1>*
    - *<"jazyk", 1>*
2. V mezifázi, která je označována jako Shuffle and Sort, se tyto záznamy přepracují do podoby *<klíč, seznam hodnot>*:
    - *<"Hadoop", [1, 1, 1]>*
    - *<"koncept", [1, 1]>*
    - *<"jazyk", [1]>*
3. V posledním Reduce kroku se pak zpracují záznamy z celého setu dokumentů předešlé Shuffle and Sort fáze, takže finální výstup může obsahovat například:
    - *<"Hadoop", 1467>*
    - *<"koncept", 653>*
    - *<"jazyk", 946>*

(Wadkar, Siddalingaiah a Venner, 2014)

Výsledek z prvního kroku obsahuje některé záznamy vícekrát a u všech je jako *hodnota* číslo 1. Toto se dá změnit odlišným naprogramováním kroku Map, takže například místo tří záznamů *<"Hadoop", 1>* by zůstal jen jeden *<"Hadoop", 3>*. Druhý způsob je však mnohem náročnější pro naprogramování, i když v některých situacích může přinést zvýšení výkonu.

(Lam, 2011)

Fáze Reduce se nemusí provádět vždy. V některých situacích může stačit pouze výstupy z fáze Map. Takové aplikace se označují jako map-only jobs.

(deRoos, 2014)

Součásti MapReduce a YARN

Ve starších verzích MR je hlavní komponenta JobTracker, která zajišťuje poměrně mnoho úkolů a to způsobuje určitá omezení. MapReduce prošel přepracováním a nová verze je označována jako NextGen MapReduce, MapReduce 2.0, MRv2, nebo také Yet Another Resource Negotiator (YARN). YARN a MapReduce však nejsou totéž. YARN převzal funkce plánovače úloh a správce výpočetních zdrojů a s MR spolupracuje (Welcome to Apache™ Hadoop®! 2014). Mezi hlavní součásti YARN se řadí ResourceManager, NodeManager a ApplicationMaster. S novou verzí MR se zvýšila flexibilita, škálovatelnost a výkon. **ResourceManager** (RM) představuje nadřízený uzel. Je to plánovač, který přiděluje výpočetní zdroje aplikacím, které je vyžadují. **NodeManager** (NM) běží na všech podřízených uzlech. Každá instance NM odesílá službě RM informace o dostupných výpočetních zdrojích na konkrétním podřízeném uzlu. Výpočetní zdroje jsou sdružovány do kontejnerů (containers), které obsahují nutné zdroje pro běh aplikace. Zdroje v kontejneru představují: CPU, operační paměť, šířka pásma (bandwidth) a místo na úložišti. Tyto kontejnery pak běží jako jednotlivé procesy na podřízeném uzlu. Každá aplikace běžící v Hadoop clusteru má přidruženou vlastní instanci **ApplicationMaster** (AM). V průběhu aplikačního výkonu odesílá AM zprávy o stavu službě RM.

(deRoos, 2014)

Input splits

Protože HDFS ukládá sobory po blocích, jejichž hranice mohou být jednotlivými soubory překročeny, používá Hadoop logickou reprezentaci souborů označovanou jako input splits. Jednotka input split je zpracovávána jednou mapovací úlohou. Aplikace MR vypočítá input splits, aby věděla, kde v bloku začíná první záznam a kde končí poslední záznam. Když poslední záznam přesahuje do jiného bloku, input splits obsahují informace o lokaci dalšího bloku a posunu bajtů

(byte offset) v řádku. V případě velkého množství zpracovávaných bloků je možné nastavit ApplicationMaster službu (nebo JobTracker v Hadoop 1), aby počítala input splits místo MR klienta, což zvýší výkon.

(deRoos, 2014)

Komprimace souborů

S dělením soborů (input splits) souvisí také jejich komprimace (komprese), která ušetří místo v úložišti. Hadoop podporuje řadu kompresních algoritmů, včetně gzip, bzip2 nebo DEFLATE. Formáty **gzip** a **DEFLATE** nepodporují dělení souborů tak, aby mohly být zpracovány paralelně, proto jsou vhodnější spíše pro menší soubory, které mohou být zpracovány jednou úlohou (podle Whitea (2012) je gzip v podstatě DEFLATE s extra záhlavím a zápatím). Formát **bzip2** podporuje dělení souborů od verze Hadoop 0.21.0, ale je relativně nevýkonný. Vhodným komprimačním nástrojem je **LZO,** který má rychlou dekompresi a podobně rychlou kompresi jako DEFLATE a zároveň podporuje vhodné rozdělení souborů.

(deRoos, 2014)

Soubor LZO musí být předzpracován indexačním nástrojem, obsaženým v knihovnách Hadoop LZO, a musí být použit odpovídající vstupní formát pro MapReduce, aby bylo možné použít dělení souborů. Distribuce Apache Hadoop neobsahuje knihovny LZO, protože jsou vázány GPL licencí a musí být opatřeny z externího zdroje. Systém může být nakonfigurován tak, aby výstupy z MR úloh byly rovnou komprimovány a při přenášení dat byl jejich objem menší.

(White, 2012)

Programování MapReduce aplikací

Programování MR úloh probíhá většinou v jazyce Java, ve kterém je také Hadoop napsaný, ale použitím Hadoop streaming je možné programovat v jakémkoliv programovacím nebo skriptovacím jazyce, který podporuje standardní vstup a výstup.

(Holmes, 2012)

Pro aplikaci je potřeba napsat tři třídy. Hlavní třída, která definuje strukturu aplikace a spouští ji, se označuje jako **driver**. Další dvě třídy se označují podle fáze zpracování, kterou vykonávají. Třída pro první mapovací krok je `Mapper` a třída pro druhý krok je `Reducer`. DeRoos (2014) používá pro ukázku data obsahující záznamy o komerčních letech v USA, podle tohoto autora jsou příklady zpracovány. Cílem aplikace je vypočítat množství letů pro každého dopravce. Třída *FlightsByCarrier.java* (driver) vypadá následovně:

```
// 1
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;

public class FlightsByCarrier {
   public static void main(String[] args) throws Exception {

      // 2
      Job job = new Job();
      job.setJarByClass(FlightsByCarrier.class);
      job.setJobName("FlightsByCarrier");

      // 3
      TextInputFormat.addInputPath(job, new Path(args[0]));
      job.setInputFormatClass(TextInputFormat.class);

      // 4
      job.setMapperClass(FlightsByCarrierMapper.class);
      job.setReducerClass(FlightsByCarrierReducer.class);

      // 5
      TextOutputFormat.setOutputPath(job, new Path(args[1]));
      job.setOutputFormatClass(TextOutputFormat.class);
      job.setOutputKeyClass(Text.class);
      job.setOutputValueClass(IntWritable.class);

      // 6
      job.waitForCompletion(true);
   }
}
```

Tato kostra driveru zůstává u všech aplikací víceméně podobná. Podle čísel v komentářích třídy je uspořádán následující popis:

1. Jsou importovány všechny potřebné Hadoop třídy. Je použito nové MapReduce API `org.apache.hadoop.mapreduce` (starší API používá balíček `org.apache.hadoop.mapred`).
2. Třída `Job` reprezentuje celou MapReduce aplikaci. Je zde nastaveno jméno třídy, která spouští úlohu a pak je nastaven identifikátor úlohy. Defaultně jsou načteny vlastnosti úlohy ze souboru *etc/hadoop/conf*, ale je možné je změnit nastavením vlastností třídy `Job`.
3. Nastaví se cesta k datům, která mají být zpracována a očekávaný formát dat. Výchozí formát vstupu je `TextInputFormat`.
4. Nastaví se třídy `Mapper` a `Reducer`. Pokud se provádí jen mapovací fáze (map-only job), nenastavuje se `Reducer` a počet úloh se vynuluje zadáním `job.setNumReduceTasks(0)`.
5. Je zadána cesta pro výstupní data, jejich formát a formát *klíče* a *hodnoty*.
6. Poslední část kódu spustí aplikaci a čeká se na potvrzovací zprávu.

(deRoos, 2014)

Další soubor definuje mapovací třídu *FlightsByCarrierMapper.java*.

```
// 1
import java.io.IOException;
import au.com.bytecode.opencsv.CSVParser;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.Mapper;

// 2
public class FlightsByCarrierMapper extends
   Mapper<LongWritable, Text, Text, IntWritable> {

   // 3
   @Override
   protected void map(LongWritable key, Text value, Context
      context) throws IOException, InterruptedException {

      // 4
      if (key.get() > 0) {
         String[] lines = new
            CSVParser().parseLine(value.toString());

         // 5
         context.write(new Text(lines[8]), new IntWritable(1));
      }
```

```
    }
}
```

Třída pro mapovací fázi se, v porovnání s ostatními, u různých aplikací nejvíce liší. Následuje její popis:

1. Jsou importovány všechny potřebné třídy. `CSVParser` se standardně mezi Hadoop třídami nenachází, ale je zde použit pro jednoduchost.
2. Explicitní nastavení `Mapper` třídy, udávající formát *klíče* a *hodnoty* na vstupu i výstupu.
3. Metoda `map` určuje názvy proměnných pro *klíče* a *hodnoty* a pro objekt `Context`, který je zapisuje.
4. V tomto bloku kódu se děje vlastní zpracování. Podmínka přeskočí separování (parsing) prvního řádku souboru, protože obsahuje jen názvy sloupců. Následně jsou jednotlivé záznamy separovány metodou `parseLine`.
5. Z pole řetězců se do objektu `Context` vrátí jako klíč devátá hodnota, která reprezentuje dopravce (podle struktury soboru) a jako hodnota číslo 1 (jeden let).

(deRoos, 2014)

Poslední třída *FlightsByCarrierReducer.java*:

```
// 1
import java.io.IOException;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.Reducer;

// 2
public class FlightsByCarrierReducer extends
   Reducer<Text, IntWritable, Text, IntWritable> {

   // 3
   @Override
   protected void reduce(Text token, Iterable<IntWritable>
      counts, Context context) throws IOException,
         InterruptedException {
      int sum = 0;
```

```
        // 4
        for (IntWritable count : counts) {
           sum += count.get();
        }

        // 5
        context.write(token, new IntWritable(sum));
    }
}
```

1. Jsou importovány potřebné třídy.
2. Specifikace `Reducer` třídy, udávající formát *klíče* a *hodnoty* na vstupu i výstupu.
3. Metoda `reduce` určuje názvy proměnných pro *klíče* a *hodnoty* a pro objekt `Context`, který je zapisuje.
4. Blok `for` cyklu sečte všechny *hodnoty* v seznamu pro daný *klíč*.
5. Metoda `write` zapíše finální součet. Jako *klíč* je použit obsah proměnné `token` a jako *hodnota* obsah proměnné `sum`.

(deRoos, 2014)

Po zkompilování kódu a vytvoření JAR souboru se může aplikace spustit následujícím příkazem:

```
hadoop jar FlightsByCarrier.jar FlightsByCarrier
<cesta_k_csv_souboru> <cesta_k_výstupnímu_soboru>
```

(deRoos, 2014)

9.3 Apache Bigtop

Hadoop je možno získat ve formě distribucí. Tyto distribuce integrují celý ekosystém projektů, týkajících se Hadoop, do jednoho celku. Jedním z takových distribucí je Apache Bigtop, který je čistě open source a zahrnuje tyto komponenty:

- Apache Crunch,
- Apache Flume,
- Apache Pig,
- Apache Solr,

- Apache Giraph,
- Apache HBase,
- Apache HCatalog
- Apache Hive,
- Apache Mahout,
- Apache Oozie,
- Apache Sqoop,
- Apache Whirr,
- Apache ZooKeeper,
- Cloudera Hue,
- LinkedIn DataFu.

Výhodou použití předpřipravené distribuce, oproti vlastnímu sestavení systému, je její snadná instalace, odpadnutí rizika nekompatibility různých verzí komponent nebo nutnosti testování. Další firmy nabízející vlastní distribuce Hadoop jsou: Cloudera, EMC, Hortonworks, IBM, Intel, MapR a další.

(deRoos, 2014)

Apache Crunch

Apache Crunch je Java knihovna, která usnadňuje vytváření jednoduchých úloh, jako spojování nebo agregaci dat. Při použití čistého MapReduce je jejich implementace zbytečně složitá. Zvláště se Crunch hodí pro zpracování dat, která neodpovídají relačnímu modelu.

(Apache Crunch, 2013)

Apache Flume

Flume slouží ke sběru, agragaci a přesunu dat z různých zdrojů a k jejich ukládání do HDFS. Zachytávaná data mohou být různého druhu, ale primárně se Flume hodí ke sběru dat z logů na webových serverech.

(deRoos, 2014)

Apache Giraph

Giraph je open source protějškem Google Pregel, založeném na paralelním výpočetním modelu BSP (Bulk Synchronous Parallel). Jedná se o nástroj pro zpracování grafů, využívaný například firmou Facebook.

(Lublinsky, Smith a Yakubovich, 2013)

Apache HBase

HBase je sloupcově orientovaná distribuovaná databáze, postavená nad HDFS. Je navržena pro možnost lineárního škálování přidáním dalších uzlů. Používá se v případech, kdy je potřeba pracovat s velkými datovými sety v reálném čase. Nejedná se o relační databázi a nepodporuje SQL, i když jsou zde určité podobnosti.

(White, 2012)

Zjednodušeně se dá říci, že se jedná o úložiště pracující s klíči a hodnotami. Klíč je kombinací označení pro řádek a sloupec v tabulce. Pokud je zadán klíč jen pro řádek, vrátí se hodnoty ze všech sloupců z tohoto řádku. Na rozdíl od typické relační databáze, kde je počet sloupců tabulky pevně dán a každý řádek musí mít stejný počet sloupců, v HBase může každý řádek obsahovat různý počet sloupců, které jsou přidávány dynamicky a jejich počet může jít až do milionů. Navíc, pokud se změní určitá hodnota v buňce, není předchozí hodnota ztracena, ale je uchovávána jako starší verze hodnoty.

(Wadkar, Siddalingaiah, Venner, 2014)

Apache HCatalog

HCatalog poskytuje jednotný a zjednodušený pohled na data uložená v HDFS. Vytváří pohled a přístup k datům podobný relační databázi. Aplikace (Pig, Hive, MapReduce) nemusí pak řešit přístup k datům příliš detailně.

(Wadkar, Siddalingaiah, Venner, 2014)

Apache Hive

Hive je nadstavba pro Hadoop, která umožňuje přistupovat k datům prostřednictvím jazyka HiveQL (Hive Query Language), který vychází z SQL (Structured Query Language). Byl vyvinut společností Facebook a uvolněn jako open source software. Dotaz v HiveQL se po spuštění automaticky přeloží do jedné nebo více MapReduce úloh a po jejich provedení se výsledky okamžitě ukazují uživateli. Tento přístup plně nenahradí přímé psaní MR úloh, ale přesto přináší velké výhody. Dovoluje provádět analýzu dat lidem, kteří by programování MR nezvládli a výsledků je dosaženo v kratším čase.

(Turkington, 2013)

Podobně jako relační datové úložiště, poskytuje Hive koncepty databází, tabulek a pohledů. Může se k němu interaktivně přistupovat přes příkazovou řádku nebo programově skrze Java Database Connectivity (JDBC). Jednoduché dotazy jsou převedeny na map-only jobs, zatímco komplexní dotazy jsou převáděny do několika úrovňových MapReduce úloh. Hive není transakční datové úložiště, nepodporuje příkazy `update`, `delete` nebo `insert` pro vložení jednoho řádku. Nejlépe se Hive hodí pro dlouho běžící dávkové dotazy, u kterých je důležitá jejich spolehlivost a doba spouštění není důležitá. Pokud může být problém vyřešen efektivně pomocí Hive, je to často preferovaný způsob před ručně psanou MapReduce úlohou.

(Wadkar, Siddalingaiah, Venner, 2014)

Apache Mahout

Mahout je knihovna algoritmů pro strojové učení, přizpůsobená pro Hadoop (Trukington, 2013).

Apache Oozie

Oozie umožňuje plánování a spouštění různých Hadoop aplikací v určitém pořadí. Tato sekvence se definuje pomocí orientovaného acyklického grafu. Vrcholy grafu představují akce a rozhodovací body, zatímco hrany ukazují pořadí a možnosti těchto akcí. Této sekvenci se říká workflow. Spuštění jednotlivých workflow může být naplánováno podle času nebo podle dostupnosti dat.

(deRoos, 2014)

Apache Pig

Společně s Hive je často zmiňován i Pig, který také slouží jako abstrakční vrstva pro psaní MapReduce úloh. Na rozdíl od Hive používá Pig jako programovací jazyk Pig Latin, který je blíže algoritmickému programování a nabízí o něco větší kontrolu nad strukturou programu. Pig podporuje komplexní datové typy (tuple, bag, map), díky kterým může pracovat s daty relačními, vnořenými, semi-strukturovanými i nestrukturovanými.

(Lam, 2011)

Pig byl vyvinut firmou Yahoo! s důrazem na jeho rozšiřitelnost, díky které mohou být jeho vnitřní funkce pro nahrávání, ukládání, filtrování, seskupování a spojování dat přepsány vlastními funkcemi (user defined functions - UDFs).

(White, 2012)

Apache Solr

Apache Solr nabízí funkcionalitu fulltextového vyhledávání, přičemž využívá knihovnu Apache Lucene jako vyhledávací stroj (search engine). Od roku 2010 jsou tyto dva projekty spojené. Lucene dokáže indexovat téměř jakákoliv data, která se dají převést na text. Kromě běžných dokumentů (Microsoft Office, PDF, OpenOffice, internetové stránky) může indexovat obrázky podle EXIF (Exchangeable Image File Format) informací nebo jiná data podle jejich metadat. Solr je nasazen například v projektech NASA Planetary Data System a JobSeeker.

(Sikora, 2012)

Apache Sqoop

Sqoop je zkratka pro „SQL to Hadoop“. Sqoop umožňuje obousměrný přenos dat mezi Hadoop a téměř jakoukoliv databází pomocí JDBC ovladače. S využitím MapReduce probíhají tyto operace paralelně, bez potřeby psát kód. Pro zvýšení výkonu je možné použít zásuvné moduly (plug-ins) pro nativní práci s různými databázemi. Sqoop přímo podporuje MySQL a PostgreSQL. Zdarma je možné sehnat moduly například pro SQL Server nebo Oracle.

(Sammer, 2012)

Apache Whirr

Podle Apache Whirr (© 2010-2013) není od března 2015 projekt nadále vyvíjen. Whirr byl vytvořen pro zjednodušení spouštění Hadoop clusterů v cloudu s využítím knihovny Apache jcloud, díky čemuž je Whirr nezávislý na poskytovateli cloudových služeb (Sammer, 2012).

Apache ZooKeeper

ZooKeeper je distribuovaná koordinační služba, která poskytuje sadu nástrojů pro vytváření distribuovaných aplikací, zvládajících částečná selhání. Těmito selháními se rozumí komunikace mezi uzly, kdy odesílatel odešle zprávu příjemci a následně nastane chyba sítě nebo jiný problém. Odesílatel pak neví, jestli příjemce zprávu dostal. Po obnovení komunikace se musí odesílatel znovu dotázat příjemce. ZooKeeper běží na více počítačích, proto je vysoce dostupný a spolehlivý.

(White, 2012)

Cloudera Hue

Hue je webové grafické rozhraní, které usnadňuje práci s různými komponentami systému Hadoop. Jedná se o HDFS, MapReduce (YARN), HBase, Hive, Pig, Sqoop, Oozie, Solr a ZooKeeper.

(deRoos, 2014)

LinkedIn DataFu

DataFu je knihovna UDF funkcí (User Defined Functions) pro Apache Pig, vytvořená ve firmě LinkedIn (Owens, Lentz a Femiano, 2013).

10 Shrnutí výsledků

Hlavním cílem práce bylo seznámení s oblastmi big data, business intelligence a data mining a ze souvisejících technologií popsat podrobněji systém Apache Hadoop. První polovina práce (kapitoly 4, 5 a 6) definuje data, typy dat a big data, popisuje jejich informační potenciál, ukazuje způsob jejich integrace a přípravy pro vykonání různých druhů reportů a procesů data mining, které pomáhají při rozhodování v různých oborech lidské činnosti. V těchto kapitolách jsou také řešeny zdroje dat a systémy pro jejich integraci, kvalita dat a možnosti řešení některých problémů. Celkově je tato část zaměřena na data a drží se spíše tradičních systému. Částečně byl hlavní cíl splněn s popisem big data a data mining. Většina dalších cílů byla také splněna, a to charakteristika dat, vysvětlení jejich významu a možnosti využití a uvedení problémů a jejich řešení při zpracování dat.

Druhá polovina práce (kapitoly 7, 8 a 9) uvádí charakteristiku business intelligece a jako názorná ukázka data mining je zde v programu IBM SPSS Modeler vytvořen model pro prediktivní analýzu, který předvádí možnost využití analýzy dat ve zdravotnictví. Dále je práce zaměřená více na moderní technologie a trendy, popisuje vlastnosti big data systému, uvádí architekturu Lambda, která splňuje podmínky pro efektivní praktické využití, její principy fungování a příklady systémů, které se v ní dají využít. Poslední kapitola je věnována technologii Apache Hadoop, které se dotýkala už kapitola předcházející. Hadoop je zde představen a podrobněji popsán, především jeho hlavní komponenty HDFS a MapReduce. Vysvětlením business intelligence a popisem systému Hadoop byl splněn hlavní cíl práce, zároveň byly příkladem v IBM SPSS Modeler a popisem Lambda architektury splněny i cíle zbývající.

11 Závěry a doporučení

Na závěr lze říci, že cíle byly celkově splněny. Obsah poskytuje informace, které umožňují vytvoření nadhledu a vlastní orientace v problematice. V některých případech je práce konkrétnější, ale většinou nezachází do přílišných podrobností, což ani nebylo zamýšleno. Vybrané téma je aktuální a celkově vyžaduje velmi rozsáhlé množství znalostí, takže i tato práce, pojatá spíše v teoretické a obecnější rovině, může přinést doplňující nebo zajímavé informace. Vzhledem k tomu lze doporučit zkoumání jednotlivých témat do větší hloubky.

Může se jednat například o podrobnější zkoumání oblasti data mining, jeho postupů, statistických metod a modelů, bližší studium potřebných technologií pro zpracování dat a jejich možností nebo porovnání jiných architektur pro big data (například Kappa architektura). Práce se nezabývá řešením ochrany citlivých dat, což má v praxi značný význam a může být dalším směrem zkoumání. Aktuální problematikou je také zpracování dat formou cloud computingu.

12 Seznam použité literatury

12.1 Tištěné zdroje

[1] AGGARWAL, Charu C. *Data mining: the textbook*. Cham: Springer, 2015. ISBN 978-3-319-14141-1.

[2] ANDERSON, Alan. *Statistics for big data for dummies*. Indianapolis, IN: Wiley, 2015. ISBN 1118940016.

[3] ČECH, Pavel a Vladimír BUREŠ. *Podniková informatika*. Hradec Králové: Gaudeamus, 2009, 232 s. ISBN 978-807-0414-798.

[4] DEROOS, Dirk. *Hadoop for dummies*. Hoboken: Wiley, 2014. ISBN 978-111-8705-032.

[5] GROSSMANN, Wilfried a Stefanie RINDERLE-MA. *Fundamentals of Business Intelligence*. Berlin: Springer-Verlag, 2015. Data-Centric Systems and Applications. ISBN 978-3-662-46530-1.

[6] HOLMES, Alex. *Hadoop in practice*. Shelter Island, NY: Manning, 2012, xxiii, 511 p. ISBN 1617290238.

[7] HURWITZ, Judith, Alan NUGENT, Fern HALPER a Marcia KAUFMAN. *Big data for dummies*. Hoboken, N.J.: Wiley, 2013. ISBN 978-1-118-64401-0.

[8] KIMBALL, Ralph a Joe Caserta. *The data warehouse ETL toolkit: practical techniques for extracting, cleaning, conforming, and delivering data*. Indianapolis, IN: Wiley, 2004. ISBN 0764579231.

[9] LAM, Chuck. *Hadoop in action*. Greenwich, Conn.: Manning Publications, 2011, xxi, 312 p. ISBN 1935182196.

[10] LAURSEN, Gert H. a Jesper THORLUND. *Business analytics: taking business intelligence beyond reporting*. Hoboken, N.J.: Wiley, 2010. Wiley and SAS business series. ISBN 978-0-470-89061-5.

[11] LUBLINSKY, Boris, Kevin T. SMITH a Alexey YAKUBOVICH. *Professional Hadoop Solutions*. Indianapolis, IN: Wiley, 2013. ISBN 978-1-118-61193-7.

[12] MANOOCHEHRI, Michael. *Data just right: introduction to large-scale data & analytics*. Upper Saddle River, NJ: Addison-Wesley, 2014. ISBN 0321898656.

[13] MARZ, Nathan a James WARREN. *Big data: principles and best practices of scalable real-time data systems*. Shelter Island: Manning, 2015. ISBN 978-1-61729-034-3.

[14] MAYER-SCHÖNBERGER, Viktor a Kenneth CUKIER. *Big Data*. Brno: Computer Press, 2014, 256 s. ISBN 978-80-251-4119-9.

[15] OWENS, Jonathan R, Jon LENTZ a Brian FEMIANO. *Hadoop real-world solutions cookbook: Realistic, simple code examples to solve problems at scale with Hadoop and related technologies*. Birmingham [England]: Packt Pub., 2013.

[16] ÖZSU, M a Patrick VALDURIEZ. *Principles of distributed database systems*. 3rd ed. New York: Springer Science+Business Media, 2011. ISBN 9781441988348.

[17] PIERSON, Lillian a Jake PORWAY. *Data science for dummies*. Hoboken, New Jersey: For Dummies, a Wiley Brand, 2015. --For dummies. ISBN 1118841557.

[18] POUR, Jan, Miloš MARYŠKA a Ota NOVOTNÝ. *Business intelligence v podnikové praxi*. Praha: Professional Publishing, 2012, 276 s. ISBN 978-80-7431-065-2.

[19] RUD, Olivia Parr. *Data Mining: praktický průvodce dolováním dat pro efektivní prodej, cílený marketing a podporu zákazníků (CRM)*. Praha: Computer Press, 2001. Rychle a jistě: Databáze. ISBN 80-722-6577-6.

[20] SAMMER, Eric. *Hadoop operations*. Sebastopol, CA: O'Reilly, 2012. ISBN 1449327052.

[21] SCHEPS, Swain. *Business intelligence for dummies*. Hoboken, N.J.: Wiley, 2008. For dummies. ISBN 978-0-470-12723-0.

[22] SKALSKÁ, Hana. *Data mining a klasifikační modely*. Hradec Králové: Gaudeamus, 2010. Recenzované monografie. ISBN 978-80-7435-088-7.

[23] TURBAN, Efraim. *Business intelligence: a managerial approach*. 2nd ed. Boston: Prentice Hall, 2011, xx, 292 p. ISBN 01-361-0066-X.

[24] TURKINGTON, Garry. *Hadoop beginner's guide: learn how to crunch big data to extract meaning from the data avalanche*. Birmingham, UK: Packt Publishing, 2013. ISBN 978-184-9517-300.

[25] WADKAR, Sameer, SIDDALINGAIAH a Jason VENNER. *Pro Apache Hadoop: Second Edition*. 2nd Edition. New York, NY: Apress, 2014. ISBN 978-1-4302-4863-7.

[26] WHITE, Tom. *Hadoop: the definitive guide*. 3rd ed. Sebastopol: O'Reilly, 2012, xxiii, 657 s. ISBN 978-1-449-31152-0.

12.2 Elektronické zdroje

[1] *Apache Crunch: Simple and Efficient MapReduce Pipelines* [online]. ©2013 [cit. 2016-02-24]. Dostupné z: http://crunch.apache.org/

[2] Apache Whirr. *The Apache Software Foundation.* [online]. © 2010-2013 [cit. 2016-03-09]. Dostupné z: http://whirr.apache.org/

[3] Apache Hadoop 2.7.1. *Hadoop.* [online]. 2015 [cit. 2016-01-08]. Dostupné z: http://hadoop.apache.org/docs/r2.7.1/hadoop-project-dist/hadoop-common/FileSystemShell.html

[4] AUGUSTIN, Jan. Správný čas na Hadoop: z laboratoře do bankovního světa. *SystemOnLine.cz: ekonomické a informační systémy v praxi* [online]. Brno, 2014 [cit. 2016-04-25]. Dostupné z: http://www.systemonline.cz/business-intelligence/spravny-cas-na-hadoop-1.htm

[5] Breast Cancer Wisconsin (Original) Data Set. *UCI Machine Learning Repository* [online]. 1992 [cit. 2016-04-20]. Dostupné z: http://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Original%29

[6] COX, M. a D. ELLSWORTH. Application-controlled demand paging for out-of-core visualization. In: *Proceedings. Visualization '97 (Cat. No. 97CB36155)* [online]. IEEE, 1997, 235-244, [cit. 2016-04-08]. DOI: 10.1109/VISUAL.1997.663888. ISBN 0-8186-8262-0. Dostupné z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=663888

[7] ČERNÝ, Michal. Big data a jejich zpracování. *Root.cz* [online]. 18. 2. 2013 [cit. 2014-06-30]. Dostupné z: http://www.root.cz/clanky/big-data-a-jejich-zpracovani/

[8] DOLÁK, Ondřej. Big data: nové způsoby zpracování a analýzy velkých objemů dat. *SystemOnLine.cz: ekonomické a informační systémy v praxi* [online]. Brno, 2011 [cit. 2014-06-30]. Dostupné z: http://www.systemonline.cz/clanky/big-data.htm

[9] DOLÁK, Ondřej. Když se řekne „Hadoop“. *Linux expres* [online]. 6. srpen 2013 [cit. 2014-06-27]. Dostupné z: http://www.linuxexpres.cz/software/kdyz-se-rekne-hadoop

[10] DOLEJŠ, Radan (ed.). *Big data: pro začátečníky a pokročilé* [online]. Praha: IDG Czech Republic, 2012 [cit. 2016-04-25]. Dostupné z: http://data.computerworld.cz/file/BigData/BigData-2012-web.pdf

[11] Hadoop: NameNode, Checkpoint Node and Backup Node. *Morris Jobke* [online]. 2013 [cit. 2016-01-11]. Dostupné z: http://morrisjobke.de/2013/12/11/Hadoop-NameNode-and-siblings/

[12] HDFS Commands Guide. *Hadoop.* [online]. 2014 [cit. 2016-01-08]. Dostupné z: http://hadoop.apache.org/docs/r2.6.0/hadoop-project-dist/hadoop-hdfs/HDFSCommands.html#dfs

[13] HDFS High Availability Using the Quorum Journal Manager. *Hadoop.* [online]. 2016 [cit. 2016-01-11]. Dostupné z: http://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-hdfs/HDFSHighAvailabilityWithQJM.html

[14] HDFS Users Guide. *Hadoop.* [online]. 2015 [cit. 2016-01-05]. Dostupné z: http://hadoop.apache.org/docs/r2.7.1/hadoop-project-dist/hadoop-hdfs/HdfsUserGuide.html#Secondary_NameNode

[15] IBM®. *IBM® SPSS® Modeler* [software]. Version 18.0. March 15, 2016. Dostupné z: http://www-01.ibm.com/software/analytics/spss/products/modeler/. Požadavky na systém: procesor x86 (Intel® Pentium®), x64 (AMD 64, EM64T), operační systém Microsoft Windows: XP Professional SP 3, Vista Enterprise SP2, Vista Business SP2, 7 Professional, 7 Enterprise, 8, 8.1, Server 2008 R2 Standard Edition, Server 2008 Standard Edition, Server 2008 R2 Enterprise Edition, Server 2008 Enterprise Edition, Server 2012 R2 Standard Edition, Server 2012 Standard Edition, Server 2012 Foundation Edition, Server 2012 R2 Essentials Edition, Server 2012 Essentials Edition, Server 2012 R2 Datacenter Edition, Server 2012 Datacenter Edition, volné místo na disku 20 GB, operační paměť 4 GB.

[16] JANČA, Pavel. Hadoop: základní nedostatky vyřešeny. *SystemOnLine.cz: ekonomické a informační systémy v praxi* [online]. Brno, 2013 [cit. 2016-01-11]. Dostupné z: http://www.systemonline.cz/clanky/hadoop-zakladni-nedostatky-vyreseny.htm

[17] Kvalitativní data. *Management mania* [online]. 2014 [cit. 2016-04-08]. Dostupné z: http://managementmania.com/cs/kvalitativni-data

[18] LUHN, H. P. A Business Intelligence System. IBM Journal of Research and Development [online]. 1958, 2(4), 314-319 [cit. 2016-04-19]. DOI: 10.1147/rd.24.0314. ISSN 0018-8646. Dostupné z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5392644

[19] MARR, Bernard. Why only one of the 5 Vs of big data really matters. *IBM Big Data & Analytics Hub* [online]. 2015 [cit. 2016-03-18]. Dostupné z: http://www.ibmbigdatahub.com/blog/why-only-one-5-vs-big-data-really-matters

[20] MCNULTY, Eileen. Understanding Big Data: The Seven V's. *Dataconomy* [online]. 2014 [cit. 2016-03-18]. Dostupné z: http://dataconomy.com/seven-vs-big-data/

[21] PETR, Pavel. *Stručný návod k ovládání IBM SPSS Statistics 19 a IBM SPSS Modeler 14* [online]. 2012 [cit. 2016-04-20]. Dostupné z: http://homel.vsb.cz/~kur138/ADDS/PetrP_IBM_Statistics_2012.pdf

[22] PRESS, Gil. A Very Short History Of Big Data. *Forbes: Tech* [online]. 2013 [cit. 2016-04-08]. Dostupné z: http://www.forbes.com/sites/gilpress/2013/05/09/a-very-short-history-of-big-data/#1f426c7055da

[23] REYNOLDS, Vince. *Big Data For Beginners: Understanding SMART Big Data, Data Mining & Data Analytics For improved Business Performance, Life Decisions & More!* [online]. 2016 [cit. 2016-03-11]. ASIN B01CBGW8KA. Dostupné z: http://www.amazon.com/Big-Data-For-Beginners-Understanding-ebook/dp/B01CBGW8KA/ref=pd_zg_rss_nr_b_549646_7

[24] RIJMENAM, Mark van. Why The 3V's Are Not Sufficient To Describe Big Data. *Datafloq: Connecting Data and People* [online]. 2013 [cit. 2016-03-18]. Dostupné z: https://datafloq.com/read/3vs-sufficient-describe-big-data/166

[25] SIKORA, Radek. *Vyhledávání v českých dokumentech pomocí Apache Solr* [online]. Brno, 2012 [cit. 2016-02-24]. Dostupné z: http://is.muni.cz/th/256499/fi_m/thesis_sikora.pdf. Diplomová práce. Masarykova univerzita, Fakulta informatiky. Vedoucí práce RNDr. Radek Ošlejšek, Ph.D.

[26] Single imputation methods. *Iris Eekhout* [online]. [2014][cit. 2016-04-18]. Dostupné z: http://www.iriseekhout.com/missing-data/missing-data-methods/imputation-methods/

[27] *Understanding metadata* [online]. Bethesda, MD: NISO Press, 2004 [cit. 2016-03-16]. ISBN 1-880124-62-9. Dostupné z: http://www.niso.org/publications/press/UnderstandingMetadata.pdf

[28] Welcome to Apache™ Hadoop®!: What Is Apache Hadoop? *Hadoop* [online]. 2014 [cit. 2016-04-01]. Dostupné z: http://hadoop.apache.org/

FIM UHK
UNIVERZITA HRADEC KRÁLOVÉ
Fakulta informatiky a managementu
Rokitanského 62, 500 03 Hradec Králové, tel: 493 331 111, fax: 493 332 235

Zadání k závěrečné práci

Jméno a příjmení studenta: **Jan Bradáč**

Obor studia: Aplikovaná informatika
Jméno a příjmení vedoucího práce: **Hana Skalská**

Název práce:
Big Data a Business Intelligence

Název práce v AJ:
Big Data and Business Intelligence

Podtitul práce:
-

Podtitul práce v AJ:
-

Cíl práce: Cílem této práce je poskytnout sjednocený přehled informací z oblastí rozsáhlých dat, business intelligence a platformy Hadoop využívanou ke zpracování dat.

Osnova práce:
1. Data a typy dat (charakteristiky, typy úloh)
2. Zdroje dat a jejich integrace pro analýzu
3. Příprava dat (transformace, missing, významové sjednocení apod.)
4. Business Intelligence, možnosti vybraného SW
5. Principy praktických řešení pro rozsáhlá data
6. Hadoop - popis a technologie

Projednáno dne: 15. 10. 2014

Podpis studenta Jan Bradáč

Podpis vedoucího práce