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

Výstupní formáty souborů dat z databáze a jejich použití v různých programovacích jazycích

Bakalářská práce

Autor: Václav, Suchánek
Studijní obor: Informační management

Vedoucí práce: Ing., Monika, Borkovcová

Hradec Králové duben 2015

Prohlášení:

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

V Hradci Králové dne 19. 4. 2015 ……..

Poděkování

Děkuji Ing. Monice Borkovcové za odborné vedení a cenné rady poskytnuté při vypracování této bakalářské práce.

Anotace

Název: Výstupní formáty souborů dat z databáze a jejich použití v různých programovacích jazycích

Práce se bude zabývat různými výstupními formáty souborů dat. V prvé řadě budou rozebírány různé formáty souborů, do kterých databáze mohou data exportovat. Jelikož takovýchto formátu existuje veliká spousta, tak budou vybrány pouze nejzákladnější z nich. Bude se jednat o formáty souborů, jako jsou: XML, SQL, CSV, PDF, Microsoft Word (pro verzi 2000 i novější) a Microsoft Excel (pro verzi 2000 i novější). V další části práce bude nastíněno, jakým způsobem se s formáty souborů dá manipulovat za pomoci programovacích jazyků Java a PHP. Pro tyto 2 jazyky bude navrhnuta aplikace, která bude generovat formáty souborů dat za pomoci daného jazyka a příslušných knihoven. Závěrem se bude vyhodnocovat, který výstupní formát by měl být použit pro exportování malého a velkého množství dat již zmíněné aplikace a který formát je nejefektivnější pro práci s daty z obecného hlediska.

Annotation

Title: Output file formats of data and its use in various programming languages

The Bachelor thesis will deal with different output formats of data files. Firstly, there will be analyzed different file formats, into which database can export the data. Because of a large amount of existing formats, will be selected only the most basic one. It will be about file formats, such as: XML, SQL, CSV, PDF, Microsoft Word (version 2000 or later) and Microsoft Excel (version 2000 or later). The following section will outline the ways of manipulating with the programming languages Java and PHP. For these two languages will be designed an application that will generate a data file formats with the help of the specific language and external libraries. In conclusion will be evaluated an optimal output format for exporting small and large amounts of data from the application, and which format is most effective for working with data from a general perspective.

Obsah

Úvod

Databázové systémy představují obrovskou a velice důležitou roli v dnešní době a bez tohoto prvku se už jen s velkou obtíží dokážou obejít různá úložiště dat. Při implementaci takového systému je nutné se zaměřit na správnou aplikaci do daného prostředí. Proto se dá tvrdit, že každý z těchto systémů se stává univerzálním a existuje zde mnoho variant provedení. Díky tomuto faktu může každý z nás narazit na velikou škálu různých formátů souborů, do kterých jsou data exportována z databází, a také na nejrůznější programovací jazyky, které mohou generovat tyto formáty za pomoci knihoven. Při výskytu takového množství prostředků si kdokoliv z nás může klást otázku: „Který formát souborů dat je nejvhodnější pro exportování dat?“ Odpověď na tuto otázku bude nastíněna právě v této práci, kde by obsažená fakta měla pomoci při rozhodování v oblasti exportu dat z databáze, či práce s velkým a malým množstvím dat.

V první části práce, při představování různých formátů výstupních souborů bude využito několika programů pro správu databází. Nejdůležitější roli zde bude mít Microsoft SQL Server 2012 a to z důvodu velice přehledné příkazové řádky SQL Query, kterou nabízí. Díky tomuto prvku se budou znázorňovat různé příklady jazyka SQL, který ponese vyexportovaná data. Navíc tento software může být realizován na desktopové bázi, neboť na počítač (na kterým je implementován) se může pohlížet jako na server, ke kterému se bude připojovat. Dalším pomocným nástrojem pro správu databáze se stane phpMyAdmin. Tento nástroj je realizován např. na serveru edu.uhk.cz a proto poslouží jako ukázka databáze, která má vzdálený přístup. V neposlední řadě bude využito programů pro správu databáze, které nemají implementováno takové množství nástrojů pro manipulaci s daty databáze a nejsou tudíž tolik náročné pro implementovaný operační systém. Bude se jednat např. o software HeidiSQL a příležitostně i další.

V další části, kde budou představeny různé manipulace exportu dat, bude využito především specializované vývojové platformy NetBeans IDE 8.0.1, která poslouží pro ukázky jazyka Java i jazyka PHP. Pro PHP bude navíc využito frameworku Nette verze 2.2.3, který poslouží pro jeho zefektivnění. U jednodušších ukázek zdrojového kódu obou jazyků se použije freewarový textový editor PSPad, popř. poznámkový blok. V další řadě bude využito localhostového serveru EasyPHP, který poslouží pro spuštění skriptů jazyka PHP v rámci webového prohlížeče Google Chrome verze 39.0.2171.95 dev-m.

Každá ukázka daného formátu obsahuje ukázkový export jedné uměle vytvořené tabulky z databáze. Tabulka je uložena na již zmíněném serveru edu.uhk.cz, byla vytvořena pomocí příkazové řádky SQL Query v phpMyAdmin. Jméno databáze: „suchava1“ s příslušným heslem bylo přiděleno při studiu databázových systémů. Tato databáze obsahuje dvě tabulky: „Osoba“ a „Adresa“, které jsou propojeny za pomoci cizího klíče. Toto schéma bylo použito jako dílčí projekt. Pro znázornění exportovaného dokumentu bude prezentována pouze postačující tabulka: „Adresa“, na které bude ukázán export její struktury a obsahu dat. V ukázkách budou vystupovat tři fiktivní adresy, které ponesou nějaká data včetně zmíněného cizího klíče.

1 Teoretická část

1.1 Formáty souborů pro export dat z databáze

1.1.1 XML

1.1.1.1 Úvodem o XML

Veliká většina dokumentů ve webovém prostředí je vytvořena pomocí značkovacího jazyka HTML (Hypertext Markup Language) a to právě z důvodu jeho jednoduchosti. Při načítání webových stránek je totiž zapotřebí, aby prohlížeč načítal co nejmenší množství dat, neboť potom se weby načítají rychleji a efektivněji. Avšak postupem času se HTML stal nedostatečným a to hlavně pro ty, kteří tvořili rozsáhlejší dynamiku webu. Značky v tomto jazyce nenesou sémantiku, neboli informace o významu dat, které obsahují. Z tohoto důvodu se do HTML zaváděly značky nové, ale ty později nepřinesly nepříliš mnoho úspěchu, protože nebyly kompatibilní pro všechny webové prohlížeče. Nakonec organizace World Wide Web Consortium (W3C) vytvořila nový jazyk XML (Extensible Markup Language), který se díky jeho struktuře a flexibilitě dal velice dobře použít. [3]

1.1.1.2 Historie XML

XML byl vyvinut z jazyka SGML (Standard Generalized Markup Language). Tento jazyk měl řešit podobné záležitosti, které řeší XML. SGML je velice složitý, specifikace SGML zahrnuje více než 150 technických stránek, je navržen pro mnoho zvláštních situací, které by mohly nastat, a pro nepravděpodobné záležitosti. Oproti těmto faktů je tento jazyk velice mocným, neboť zahrnuje spoustu užitečných nástrojů. Kvůli složitosti se v 90. letech vyvíjela „odlehčená“ verze SGML, specifické pojmenování již napovídá, že se jednalo především o zjednodušení. Hlavním cílem bylo zachování silných stránek tohoto jazyka a odstranění nadbytečných a nijak užitečných prvků. V únoru roku 1998 vznikl XML 1.0, který dosahoval velice pozitivního hodnocení. V dalších fázích vývoje se objevily jmenné prostory (XML namespaces), rozšiřitelný jazyk pro stylové šablony XSL (Extensible Stylesheet Language), dále se XSL vyvinul na XSL Transformations (XSLT) a XSL Formating Objects (XSL-FO). V době postupu XSL byl již používán kaskádový stylový jazyk CSS (Cascading Stylesheet Language), který byl využíván u jazyka HTML, ale byl i velice dobře použitelným pro XML. Z toho důvodu bylo CSS úrovně 2 záměrně navrženo pro stylové dokumenty v XML a tím se tento jazyk stal stejně důležitým jako HTML. Pro propojení XML dokumentů do hypertextové sítě se zrodil rozšiřitelný odkazovací jazyk XLL (Extensible Linking Language).

Dále následovaly standardy tohoto jazyka: XLink a XPointer, jenž první z nich popisuje spojení mezi dokumenty a druhý adresuje jednotlivé části XML dokumentu. Později se stal objektem zájmu rozhraní pro přístup k obsahu XML dokumentu z programů sestavených za pomoci jazyků Java, Javascript nebo C++. Pro tuto činnost vzniklo nejjednodušší API (Application Programming Interface), které definovalo daný dokument jako objekt, který obsahoval další objekty. Kompatibilita se stala hlavním kritériem při vytváření jednotlivých rozhraní, proto vzniklo jednoduché API pro XML, pojmenované SAX. Později v roce 2000 se z něho vyvinul SAX2, který měl propracovanější možnosti ohledně konfigurace a dalších věcí. Vyjma výše zmíněných prvků se v moderní době vyvíjí rozšiřování základní specifikace XML. Jedná se o prvky různého charakteru: XFragment, XML Schemas, XHTML, XML Query Language, Canonical XML, XML Signatures a mnoho dalších rozšíření jazyka XML. [4]

1.1.1.3 Použití

V XML hodně podobně jako v HTML se využívá párových tagů. Každý dokument obsahuje jeden kořenový (root) tag a všechny ostatní se do něj zanořují. Výsledná podoba tvoří stromovou strukturu s libovolně velkým množství elementů. Pro přiblížení bude nastíněn rozdíl krátkého kódu v HTML i v XML.

Ukázka v HTML vypadá následovně:

```
<span>Automobil <b>Mustang 1967</b> cena 121 000 Kč</span>
```

Ukázka v XML:

```
<automobil>
  <značka>Mustang</značka>
  <rok>1967</rok>
  <cena mena=''CZK''>121000</cena>
</automobil>
```

[23]

Z ukázky je možné si povšimnout, že XML je rozsáhlejší, než HTML, zahrnuje více informací a je celkově přehlednější. Nyní už je zřejmé, jakým způsobem by se s XML mělo pracovat, pro dobrou představu o exportovaném souboru tohoto formátu poslouží ukázka exportu dat databáze, která je umístěna v přílohách této práce.

1.1.1.4 DTD

1.1.1.4.1 Úvodem o DTD

DTD (Document Type Definition) je starším, ale za to často využívaným systémem pravidel. Každý dokument či atribut ve spojitosti s XML dokumentem má stanovena pravidla právě podle DTD. Pokud tomu není tak učiněno, tak poté dokument nebude validní. Jednotlivá pravidla se dají psát pomocí již zmíněného systému DTD anebo systému XML Schema. [1]

1.1.1.4.2 Vytvoření DTD

Nejjednodušším způsobem pro psaní tohoto systému pravidel je využití interního DTD, který se tvoří uvnitř daných XML dokumentů. V dokumentu se pro zahájení používá: „`<!DOCTYPE root [`“, z čehož „`root`“ označuje kořenový element DTD definice daného dokumentu. Znak ukončující definici pravidel vypadá následovně: „`]>`“. Mezi tímto zahájením a ukončením bude prostor pro obsah definice typu dokumentu, který bude zmíněn v následující kapitole. [1]

V případě, že je potřeba definovat pravidla pro více souvisejících dokumentů, tak se využívá externího DTD. Pro využití je nutné vytvořit si externí soubor, který ponese sadu pravidel. Poté v tomto souboru je možné snadno odkázat na každý příslušný XML dokument pomocí URL adresy. V konečné fázi je potřeba daný soubor uložit jako text a přidělit mu specifickou příponu: „.dtd“. [1]

1.1.1.4.3 Vytvoření obsahu DTD

1.1.1.4.3.1 Vytvoření elementu

Tvorba obsahu je shodná jak pro psaní interního, tak pro psaní externího DTD. Je nutné, aby definice zahrnovala veškeré elementy (jejich obsah a strukturu), které se nacházejí v XML dokumentu. Definovat příslušný element je možné pomocí: „`<!ELEMENT tag`“, z čehož „`tag`“ označuje název definovaného elementu. Dále bude následovat vhodné klíčové slovo pro specifikaci obsahu. Pokud bude zadáno: „`EMPTY`“, tak bude očekáváno, že daný element nenabývá obsahu. Pro definování jakékoliv kombinace nespecifikovaných elementů a textu se využívá: „`ANY`“. Pokud je potřeba určit daný obsah pro příslušný element, tak se používá: „`(contents)`“. Parametr v závorce představuje název popisovaného elementu nebo textu (je možné popsat oba prvky zároveň), který bude příslušný element zahrnovat. Pro zakončení definice příslušného elementu se použije symbolu: „>“, který slouží pro vyznačení konce deklarace. [1]

1.1.1.4.3.2 Definice obsahu elementu

Dále následuje definice obsahu elementu. U elementů by mělo být dáno, jaký typ obsahu zahrnují. K samotné specifikaci se využívá manipulace s již zmíněným: „`(contents)`“. Obsah elementu lze definovat pouze jako text. K tomu se užívá: „`(#PCDATA)`“. Dále je možné specifikovat potomka (element obsažený v příslušném elementu) pomocí: „`(child)`“, kde „`child`“ označuje název potomka. Těchto vnořených elementů je možné definovat i více. Pro představu poslouží tato ukázka: „`(child1, child2, child3)`“. Dále existují definice voleb, které jsou indikovány pomocí specifických symbolů. Např. symbol: „|“, který zajistí výskyt druhého elementu, pokud se neobjeví element první. Ukázkou může být: „`(child1 | child2)`“. V další fázi je možné určit počet jednotek daných elementů. K tomu se využívají 3 následující symboly: „?“ (indikace možnosti alespoň jednoho výskytu jednotky v dokumentu), „+“ (pro specifikaci nutnosti alespoň jednoho výskytu v dokumentu) a „*“ (indikace pro možnost výskytu jednotky tolikrát, kolikrát bude zapotřebí nebo ani jednou v definovaném dokumentu). Pro představu užití těchto 3 symbolů následuje ukázka, jejíž obsah by měl být zřejmý z předešlého textu:

„`<!ELEMENT animal (name+, threats, weight?, lenght?, source, picture, subspecies*)`“

[1]

1.1.1.4.3.3 Definice atributů elementu

Atributy slouží pro přidání doplňkových dat k elementu. Je to užitečnější metoda, než rozdělování příslušného elementu na menší řetězce informací. Nejprve je zapotřebí definovat atribut v DTD a poté může být použit v samotném obsahu XML dokumentu. Z hlediska definice pravidel se atribut může zahrnout následovně: „`<!ATTLIST population year CDATA #IMPLIED>`“, kde „`population`“ představuje název elementu, „`year`“ definuje atribut (název informace, která je vyžadována pro přidání do elementu, „`CDATA`“ označuje specifikaci pro hodnotu atributu (v tomto případě se bude skládat z nějaké kombinace znaků) a nakonec „`#IMPLIED`“ vypovídá, že daný element bude obsahovat volitelný atribut „`year`“. Po této definici je možné atributu využít v obsahu dokumentu. Pokud se bude vycházet z předchozího příkladu, tak by užití mohlo vypadat následovně: „`<population year=’1999’>445</population>`“. Příklad nese informaci o příslušném počtu populace ve specifickém roce 1999. Pokud by atributu použito nebylo, tak by se tyto informace vyjádřily pomocí rozdělení

elementu na menší řetězce informací. Pro představu tohoto rozdělení poslouží následující ukázka, která vychází z dat ukázek předchozích:

```
<population>
    <year>1999</year>
    <quantity>445</quantity>
</population>
```

[1]

1.1.1.4.4 Příklady DTD

V následující ukázce je možné si povšimnout interního zápisu DTD společně se specifickým obsahem. Jednotlivé elementy jsou definovány v rámci systému pravidel a poté jsou použity. Zde již zmíněný příklad:

```
<?xml version="1.0"?>
<!DOCTYPE note [
<!ELEMENT note (to,from,heading,body)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body (#PCDATA)>
]>
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend</body>
</note>
```

[22]

Další příklad bude znázorňovat externí zápis DTD. Resp. obsahová část těchto pravidel zde bude chybět (bude totiž vyčleněna do externího souboru, na který bude odkázáno z daného XML dokumentu). Následuje ukázka dokumentu:

```
<?xml version="1.0"?>
<!DOCTYPE note SYSTEM "note.dtd">
<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

[22]

A zde bude znázorněn příklad externího souboru, který vychází z předchozí ukázky:

```
<!ELEMENT note (to,from,heading,body)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body (#PCDATA)>
```

[22]

1.1.1.5 Výhody XML

Největším přínosem jazyka XML je možnost použití vlastních párových značek (tagů) při práci v dokumentech. Pokud je pozornost obrácena na jednu z předchozích kapitol, která polemizuje o použití, tak je možné zaznamenat, že data jsou vhodně strukturována a tento fakt velice podporuje vyhledávání a samotnou orientaci. Dále je možné XML nalézt v mnoha oblastech. Jedná se např. o webové stránky, uplatnění v elektronickém publikování, při výměně dat mezi systémy odlišného charakteru, apod. [5] Tento jazyk nabízí velice dobrou možnost přenášení dat. Je možné, že dokument vytvořený specifickým programem, který je realizován pomocí nějaké platformy, by nemusel být čitelný v jiném programu realizovaném na platformě odlišné. Proto se může využít právě XML, který je čitelný v jakémkoliv programu, který dokáže číst textové soubory. Jelikož jsou data i značkování uloženy v podobě textu, tak není nutné se zabývat doplněním každého osmého bajtu, nemusí se brát ohled na složení čtyřbajtového čísla ze dvou komplementárních celých čísel nebo z desetinného čísla podle IEEE 754, apod. Díky flexibilitě tohoto jazyka je zde prostor po využití XML jak vývojáři, tak autory dokumentů. XML představuje tzv. meta-značkovací jazyk, což znamená, že nedefinuje pevně stanovené značky a elementy, a proto tyto prvky lze použít v různorodých oblastech. Např. realitní agentury mohou využívat prvků, které budou definovat byty, nájmy, lokality. Dalším příkladem může být oblast chemie. Zde obdobným způsobem mohou prvky popisovat atomy, vazby, reakce, apod. [4] Z příkladů je zjevné, že XML slouží pro vytváření dalších jazyků. Poslouží k vytváření individuálního uživatelského jazyka a poté je jej možné použít k zformátování vlastních dokumentů. Poté se tento jazyk dá označovat jako uživatelský značkovací jazyk, který se obvykle nazývá XML aplikace. [1]

1.1.1.6 Nevýhody XML

Tento jazyk je mnohdy považován za striktní. Přesně definuje, kde se značky v dokumentu mohou vyskytovat, jak se k elementům přiřazují atributy apod. Existují zde analyzátory, které jsou schopné čtení a porozumění jakémukoliv dokumentu vytvořeného pomocí XML. Správně formulovanými dokumenty jsou označovány ty, které jsou čitelné pomocí analyzátoru. Při výskytu chyb ve formulaci elementů v XML dokumentu procesor XML jednoduše odmítne daný soubor zpracovat. Dále tento jazyk není programovacím jazykem a neexistuje v něm nic takového, jako je kompilátor kódu, který by prošel soubory v XML a poté vygeneroval spustitelný kód. XML nezprostředkovává přenos dat po síti, neboť se nejedná o síťový protokol. K přenosu dat se obvykle využívá HTTP, FTP, NFS a další jiné síťové protokoly. Dále tento jazyk nepředstavuje databázi. Data v XML se dají do databází ukládat, ale samotná databáze nemůže být reprezentována jako XML dokument. [4]

1.1.2 SQL

1.1.2.1 Úvodem o SQL

SQL (Structured Query Language) je jazyk pro komunikaci s databází. Tento jazyk je velice často využíván databázovými systémy, jako jsou: Oracle, Sybase, Microsoft SQL Server, Access, Ingres a další. [17]

Jazyk se hlavně používá pro vytváření databází a jejich struktur. V tomto případě se pracuje s Data Definition Language (DDL), který nabízí příslušné operátory. Dále se SQL specializuje na Data Manipulation Language (DML), kde jsou uživateli nabídnuty čtyři základní operace s daty: čtení, zápis, mazání a upravování. [2]

1.1.2.2 Vlastnosti

Tento jazyk vyniká především jeho jednoduchostí. Příkazy jsou tvořeny z anglických slov, které nesou přímo samotný význam operací. Sekvence operátorů jsou co nejkratší a navrženy tak, aby byly snadno zapamatovatelné. SQL je neprocedurální, proto se do příkazů píší pouze „věci“, které jsou uživatelem požadovány, a není specifikována cesta, jak těchto dat dosáhnout. Dále je tento jazyk velice přizpůsobivý. Nezáleží na nadbytečných mezerách v příkazech, velká a malá písmena nejsou striktně kotvena, nezáleží na stylu použití uvozovek, apod. Je dobře využitelný jak pro uživatele začátečníky, tak pro pokročilé. [3]

1.1.2.3 DML a jeho použití

Tato oblast jazyka SQL zahrnuje čtyři základní CRUD operace s daty databáze. Tato zkratka je složena z anglických slov: create, read, update a delete. [2] Nejprve bude rozebráno „create“. V překladu znamená: „vytvořit“. Pro vytváření nových dat v databázích SQL používá příkaz „`INSERT`“. Ten způsobí, že do tabulky bude vložen nový záznam. Vložení nových dat do tabulky „`zam`“ může vypadat takto:

```
INSERT INTO zam VALUES ('Petrásek', 2, 6200)
```

[20]

Další je slovo „read“, které v českém jazyce znamená: „číst“. SQL využívá jednoduchý příkaz „`SELECT`“, díky kterému se dají vybrat specifikovaná data z určité tabulky. Většinou se vybírají všechna data, která jsou obsažena v tabulce. Na to se používá symbol: „*“, pokud ale uživatel chce vybrat pouze určitá data tak za příkaz přidá omezení pomocí: „`WHERE`“, kde dále specifikuje podmínku vyhledávání. [2] Následuje ukázka, v první řádku je ukázán příkaz pro výběr všech dat z tabulky a ve druhém pouze některých. Výběr dat z tabulky „`zam`“ může vypadat takto:

```
SELECT * FROM zam
SELECT * FROM zam WHERE oddcis = 1
```

[20]

Třetí v pořadí je „update“ s významem: „upravit“, někdy i: „aktualizovat“. Stejným způsobem je nazván i příkaz: „`UPDATE`“, který má za úkol vybrat určitá data a jejich staré hodnoty nahradit hodnotami novými. [2] Podobně jako v předchozím případě se dá specifikovat určitá množina dat anebo se označí všechna data ve vybraném sloupci. Takto může být použit příkaz „`UPDATE`“, který upraví určitou množinu dat:

```
UPDATE zam SET plat = 1.2 * plat WHERE oddcis = 2
```

[20]

V poslední řadě bude znázorněno mazání dat: „`DELETE`“. Tento příkaz funguje podobným způsobem jako „`UPDATE`“, můžeme tedy specifikovat určitou množinu dat, která má být z databáze odstraněna, ale i nemusíme. [2] Ukázka vymazání některých dat z tabulky „`zam`“:

```
DELETE FROM zam WHERE oddcis = 2
```

[20]

Zde byly znázorněny metody pro práci s daty databáze. Existuje ještě více možností pro práci s daty, avšak v této práci budou postačující pouze tyto základní.

1.1.2.4 DDL a jeho použití

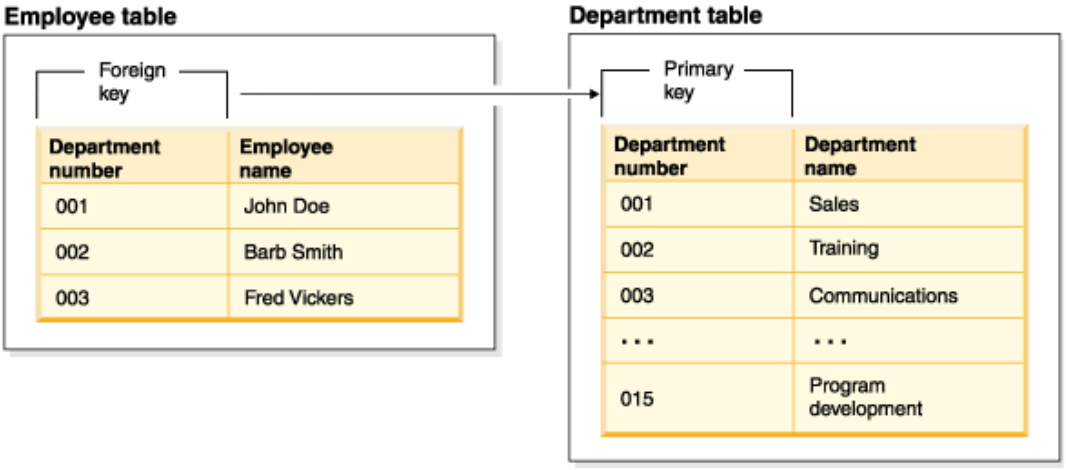
Tato sekce je o něco obtížnější, než DML, protože se zde nachází více operátorů, ze kterých není ihned zřejmé, jakou funkci přesně plní. DDL se používá k vytváření struktury databáze. Pod pojmem struktura je ukryto mnoho elementů. Součástí struktury databáze může být: tabulka, doména, schéma, pohledy, indexy, omezení a mnoho dalších. Vzhledem k ukázce budou rozebrány pouze klíčové prvky, nezbytné pro pochopení názorného exportu databáze. [2]

Základními příkazy DDL jsou: „`CREATE`, `ALTER` a `DROP`“. Používají se k vytváření, úpravám a mazání strukturních částí dat databáze. Nejprve bude znázorněno smazání tabulky, včetně všech podřazených tabulek, které jsou propojeny za pomoci cizího klíče (pro tuto možnost se používá možnost: „`CASCADE`“):

```
DROP TABLE TableName [RESTRICT | CASCADE]
```

[2]

Další příkaz „`ALTER TABLE`“ slouží k šesti základním operacím. První dva z nich jsou: přidání nového sloupce do tabulky a odebrání sloupce z tabulky. Mezi další základní součásti DDL patří operace pro práci s omezeními. Tyto omezení jsou dobré k propojování tabulek. V této oblasti je použito cizích klíčů, které provazují tabulky mezi sebou. Návaznost je zprostředkována za pomoci dvou sloupečků ze dvou rozdílných tabulek. Na jedné straně se nachází tabulka se sloupcem nastaveným jako primární klíč a druhá tabulka se sloupcem nesoucím index cizího klíče. Pokud jsou tyto dvě tabulky provázány z hlediska DML, tak sloupce zpravidla obsahují stejné hodnoty. [2]



Obrázek 1: Cizí klíč [13]

K posledním úpravám v příkazu „ALTER TABLE“ patří nastavení a zrušení výchozího sloupce. To zapříčiní vložení výchozí hodnoty do sloupce, pokud v „insertu“ uživatel nezadá konkrétní hodnotu do daného sloupce a nechá ji nevyplněnu. Většinou se v takovém to sloupci objeví hodnota: „NULL“, neboli hodnota prázdná. [19] Nyní následují ukázky všech šesti zmíněných operací:

```
ALTER TABLE Osoba ADD COLUMN telefon
ALTER TABLE Osoba DROP COLUMN prijmeni
ALTER TABLE Osoba ADD CONSTRAINT nazev_ciziho_klice FOREIGN KEY (adresa_id)
REFERENCES Adresa (adresa_id)
ALTER TABLE Osoba DROP CONSTRAINT nazev_ciziho_klice RESTRICT/CASCADE
ALTER TABLE Osoba ALTER telefon SET DEFAULT null
ALTER TABLE Osoba ALTER telefon DROP DEFAULT
```

[vlastní zpracování][1]

Nakonec bude rozebrán příkaz: „CREATE“, který se dá označit jako klíčový a velice podstatný, neboť veliký význam nabývá právě při exportování databází. Využívá se především při vytváření tabulek pomocí příkazu: „CREATE TABLE nazev_tabulky ()“, kde se dále specifikují její parametry. Mezi parametry se dá zařadit veliká spousta elementů, jako např.: datové typy, názvy sloupců, výchozí hodnoty, primární klíč, cizí klíče, možnost provedení akce při úpravě nebo mazání a mnohem více. [2]

Datových typů je celá řada a různými verzemi jazyka SQL také přicházejí změny v deklaracích datových typů. Jedná se o charakterizaci znaků a symbolů, které se mohou ve sloupci vyskytovat. Pokud do sloupce přijde jiný datový typ, než ten, který je nastaven, tak databáze vypíše chybové hlášení, popř. se ve sloupci nastaví hodnota na „null“. Datové typy chrání konzistenci dat a také může chránit databázi před útoky hackerů, neboli před „neočekávaným“ typem dat. [2] Vzhledem k velkému množství datových typů budou znázorněny jen některé z nich. Následující tabulka je pro představu toho, jakým způsobem jsou deklarovány datové typy v jazyce SQL.

[1] Upraveno dle CONNOLLY, Thomas M a Carolyn E BEGG, 2005, s. 173

Datový typ	Deklarace
boolean	BOOLEAN
character	CHAR, VARCHAR
bit	BIT
exact numeric	NUMERIC, DECIMAL, INTEGER, SMALLINT
approximate numeric	FLOAT, REAL, DOUBLE PRECISION
datetime	DATE, TIME, TIMESTAMP
interval	INTERVAL
large objects	CHARACTER LARGE OBJECT, BINARY LARGE OBJECT

Tabulka 1: Datové typy [vlastní zpracování][2]

Následuje ukázka vytvoření schématu tabulky „`Osoba`“:

```
CREATE TABLE IF NOT EXISTS `Osoba` (
 `osoba_id` int(11) NOT NULL AUTO_INCREMENT,
  `jmeno` varchar(15) COLLATE utf8_czech_ci NOT NULL,
  `prijmeni` varchar(30) COLLATE utf8_czech_ci NOT NULL,
  `datum_narozeni` date DEFAULT NULL,
  `telefon` char(9) COLLATE utf8_czech_ci DEFAULT NULL,
  `adresa_id` int(11) NOT NULL,
  PRIMARY KEY (`osoba_id`),
  KEY `klic` (`adresa_id`)
)
```

[vlastní zpracování]

Obdobným způsobem se dá vytvořit celá databáze pomocí příkazu: „`CREATE DATABASE`“, kde se dále specifikují parametry (znaková sada a další). [15] Avšak pro tuto práci budou postačující základy sekce DDL, která byla nastíněna. Nyní by čtenář měl mít dobrou představu o použití jazyka SQL a ukázka pro export dat do souboru SQL je obsažena v přílohách této práce.

[2] Upraveno dle CONNOLLY, Thomas M a Carolyn E BEGG, 2005, s. 159

1.1.3 PDF

1.1.3.1 Úvodem o PDF

Formát PDF (Portable Document Format) byl vytvořen v roce 1993 firmou Adobe Systems. Je určen především pro elektronickou výměnu dokumentů a jejich tisk. [6] Manipulace s těmito dokumenty jsou značně omezeny. PDF dokumenty jsou vytvořeny pro náhled jejich obsahu, pro navigaci po těle dokumentu, pro tisknutí a pro přeposlání na jiné zařízení. Soubor může obsahovat textové informace, obrázky (rastrové i vektorové), tabulky a jiné elementy. [12] Po konvertování do PDF souboru většinou nejde proces vrátit zpět a následně obsah upravit. To může být výhodou i nevýhodou, záleží na situaci. Tento formát je využívat především pro magazínové články, produktové brožury, návody, různí zadání a další. [24]

1.1.3.2 Vlastnosti

Nejvýznamnější charakteristika formátu PDF je fakt, že zachovává integritu souboru. Velice často se stává, že pokud konvertujeme data mezi rozdílnými formáty nebo je jednoduše převádíme z psané podoby do elektronické, tak se obsah graficky změní nepřívětivým způsobem. Tento formát však zcela zachovává vzhled a dokonce i rozlišení. PDF je spolehlivý. Nepotřebuje složitý software pro otevření a rozdílné programy jsou schopny otevřít PDF dokumenty stejně i na různých zařízeních. Hodně využívaným je v dnešní době např. Adobe Acrobat Reader. Soubory se řídí podle normy ISO 32000, která zaručuje jejich integritu a dlouhodobou životnost. Dalším důležitým rysem tohoto formátu je důvěryhodnost. Dokumenty zahrnují ochranu pomocí hesla, která může bránit k jejich přístupu, kopírování, tisknutí nebo upravování. Díky tomu je PDF formát hodně využíván podniky a státními organizacemi. Je podporován velikou škálou platforem. Soubor si uživatel může otevřít v operačních systémech, jako jsou: Windows, Mac OS, Android (u mobilních telefonů) a IOS (iPhone, iPad). Dalším přínosem je možnost elektronických podpisů. Stačí k tomu mít bezplatný software Adobe Reader XI nebo Adobe Reader pro mobilní zařízení. [6]

1.1.3.3 Struktura jednoduchého dokumentu

V případě, že se bere v úvahu soubor PDF o menším obsahu dat, tak překvapivě musí nést mnoho elementů, které tato data dále zpracovávají. Jednoduchý PDF dokument se skládá ze 4 základních sekcí. První z nich je tzv. „trailer dictionary“, který poskytuje informace o tom, jakým způsobem se dají číst ostatní objekty, které jsou v souboru obsaženy. Další sekcí je dokumentový katalog tvořící kořenový prvek v objektovém diagramu. Třetí z nich je tzv. „page tree“, který má na starosti stanovení všech stránek dokumentu. Poslední sekce představuje pravidlo, které vypovídá o tom, že každá strana dokumentu musí nabývat nějakého vlastního obsahu, který zahrnuje instrukce pro vykreslování textu a grafiky dané stránky, a dále že každá stránka musí mít své vlastní zdroje zahrnující např. fonty písma. [25]

1.1.3.4 Zápis PDF

Zápis PDF dokumentu je mnohem jednodušší, než jeho samotné čtení. To je především z toho důvodu, že při čtení takového dokumentu je potřeba převést příslušné byty na grafické objekty a při procházení se zaznamenávají úplně všechny prvky celého dokumentu. Dokonce se vyhledávají i takové, které se v souboru mohou potencionálně ukrývat. Zápis dat do PDF je podstatně rychlejší, neboť stačí použít pouze podmnožinu dat, která je v dokumentu obsažena. Postup, při kterém dochází ke konvertování bytů na grafické objekty, vypadá následovně:

1. Výstup záhlaví dokumentu.
2. Odstranění všech objektů, které žádným způsobem nesouvisí s jinými PDF objekty. Tento krok zamezuje zapisování objektů, které nejsou potřebné.
3. Rekalkulace objektů. Provede se kontrolní součet všech objektů a poté budou indexovány od 1 do n.
4. Postupné vypisování objektů. Pořadí je dáno příslušnou indexací a při procesu je každý byte offset zaznamenán do relační databáze.
5. Kompletní sestavení relační databáze.
6. Vytvoření prvků: „trailer“ a „trailer dictionary“, které byly již zmíněny u struktury dokumentu PDF, a ještě je sestaven tzv. „end-of-file marker“, který jednoznačně určuje konec daného souboru. [25]

1.1.3.5 Ukázkový příklad exportu tabulky adresa v PDF

Databáze: suchava1, Tabulka: Adresa

adresa_id	ulice	cp	mesto	psc
7	Kroměřížská	23	Jíčín	50401
8	U Jelena	26	Pardubice	50401
10	Družstevní	113	Nový Bydžov	50401

Tabulka 2: Tabulka v PDF [vlastní zpracování]

1.1.4 CSV

1.1.4.1 Charakteristika CSV

CSV (Comma-separated values) je formát souboru založen na základních konceptech. Primárně je určen pro tabulkovou výměnu dat, proto se uživatel může setkat s exportováním dat do souboru nazvaného: „CSV pro MS Excel“.

Podoba tohoto formátu je následující: soubor obsahuje text, který je rozdělen do řádků – tyto řádky odpovídají formátu možné budoucí tabulky. Dále jednotlivé položky textu jsou odděleny čárkou a hodnoty položek jsou obvykle uzavřené do uvozovek. Díky této primitivní struktuře se formát stává jednoduchým a nenáročným pro program, ve kterém je otevřen. Další jeho velikou výhodou je snadná čitelnost. CSV je možné otevřít ve většině textových editorů a tabulkových procesorech, pro které je mj. primárně určen [8].

1.1.4.2 Ukázkový příklad exportu tabulky adresa v CSV

```
"7","Kroměřížská","23","Jíčín","50401"
"8","U Jelena","26","Pardubice","50401"
"10","Družstevní","113","Nový Bydžov","50401"
```

[vlastní zpracování]

1.1.5 Microsoft Word

1.1.5.1 Charakteristika Microsoft Word

Jedná se o textový editor s celou řadou nástrojů a možností manipulace textu. Nabízí: otevírání, ukládání, zavírání, sdílení textových dokumentů a další. Čím novější verze tohoto produktu, tak tím přibývá i více prvků a možností. Nejnovější verzí v současné době je: Microsoft Word 2013, který je součástí balíčku Microsoft Office 2013. Pro účely této práce byla využita verze 2000 a novější, neboť byla dostupná její licence. [7]

Navigace tohoto editoru se skládá ze tří základních částí. První je navigační panel pro práci s obsahem dokumentu, který obsahuje panely nástrojů. Tato sekce je nejvyužívanější a nejvíce frekventovaná, protože uživatel zde nastavuje veškerou sémantiku textu. Může se zde: nastavit velikost, barva, styl písma; členit text do odstavců; vytvářet tabulky a nové styly písma; nastavovat parametry celého dokumentu a mnohem více. Další částí je tzv. „backstage view“. Tento prvek se velice zdokonaloval s novějšími verzemi tohoto produktu. Jedná se o možnost souboru, kde uživatel může spravovat a nahlížet na informace o právech uživatelů, podrobnosti o úpravách dokumentu, poslední zaznamenaná úprava, vlastnosti dokumentu apod. Třetí část zahrnuje základní operace s dokumentem. Jsou to především operace jako: otevírání, zavírání a ukládání dokumentu. [7]

1.1.5.2 Ukázkový příklad exportu tabulky adresa v Dokument Word (*.docx)

```
Databáze suchava1
Struktura tabulky Adresa
Pole      Typ         Nulový Výchozí
adresa_id int(11)     Ne
ulice     varchar(30) Ano    NULL
cp        varchar(10) Ano    NULL
mesto     varchar(30) Ano    NULL
psc       char(5)     Ano    NULL
Vypisuji data pro tabulku Adresa
7  Kroměřížská 23  Jíčín       50401
8  U Jelena    26  Pardubice   50401
10 Družstevní  113 Nový Bydžov 50401
```

[vlastní zpracování]

1.1.6 Microsoft Excel

1.1.6.1 Charakteristika Microsoft Excel

Jedná se o tabulkový procesor, vytvořený firmou Microsoft. Nejnovější verzí v současné době je: Microsoft Excel 2013, který je součástí balíčku Microsoft Office 2013. Opět existuje veliká řada verzí tohoto produktu. Zde už musí být uživatel více obezřetný, než u verzí produktu Microsoft Word, neboť Excel obsahuje specializované výpočetní funkce, které se mohou s odlišnou verzí měnit (v jejich použití). Dále se mění grafické uživatelské prostředí, ale to je velice podobné textovému editoru Microsoft Word, proto nebude více rozebíráno. [11]

Jak už bylo nastíněno, Microsoft Excel se liší od produktu Microsoft Word tím, že není zaměřen na úpravu grafické oblasti textu, ale jak už napovídá označení „procesor“, tento produkt je navržen pro práci s daty. Pro účely této práce bude využito verze 2000 a novější, neboť byla dostupná její licence. V Microsoft Excel 2010 je spousta metod a nástrojů, pomocí kterých se dá s daty pracovat. Celá struktura dokumentu je složena z buněk, které může uživatel využívat libovolným způsobem. Navíc jednotlivé buňky mají možnost využít nějakou výpočetní funkci, která je obsažena v tomto produktu. Může se jednat o funkci součtu několika vybraných buněk. Znázornění takovéto funkce může být následující:

```
=SUMA(H2:H6)
```

[11]

1.1.6.2 Ukázkový příklad exportu tabulky adresa v Dokument Excel

7	Kroměřížská	23	Jičín	50401
8	U Jelena	26	Pardubice	50401
10	Družstevní	113	Nový Bydžov	50401

Tabulka 3: Export v MS Excel [vlastní zpracování]

1.2 Programovací jazyky pro generování různých formátů souborů dat

1.2.1 PHP

1.2.1.1 Úvodem o PHP

PHP (Hypertext Preprocessor) je programovací jazyk, který slouží k vytváření dynamických (interaktivních) webových stránek. Je tvořen ze skriptů, které jsou načítány do HTML stránek. Tyto skripty jsou načítány z hlediska architektury client-server procedurálně (od shora dolů). Byl vyvinut společností The PHP Group a nejnovější aktuální verzí je PHP 5.6.1. [16]

Dynamická (interaktivní) stránka je webová stránka, která při každém navštívení může vypadat jinak. Příkladem může být změna data a času, které se na stránce vyskytují. Dále stránka sama o sobě neslouží právě jenom pro prohlížení (fotografií, článků, apod.), ale obsahuje navíc nějaké funkce. Nejčastěji to bývá formulář emailu, jehož kostra je sice napsána za pomoci jazyka HTML, ale potvrzení (submit) formuláře může už být zprostředkováno za pomoci nějakého interaktivního jazyka, třeba již zmíněného PHP. [9]

1.2.1.2 Popis zahájení PHP skriptu

Ze začátku jakýkoliv uživatel zašle požadavek na server tím, že klikne na link nebo použije příslušnou URL adresu. Webový server rozpozná, zda zaslaná adresa URL je nebo není skriptem jazyka PHP. V pozitivním případě, server uvede PHP engine do aktivního stavu a skript se spustí. Interpret PHP skriptu začne procházet skript sekvenčním (postupným) způsobem a začnou se vykonávat příkazy skriptu. V momentě, kdy interpret dokončí čtení, tak se dokončí utváření HTML stránky a její kód je poslán na stranu klienta. Tam je kód přečten a zobrazen za pomoci internetového prohlížeče. [9]

1.2.1.3 Ukázka PHP skriptu

Každý skript jazyka PHP lze poznat jednoduchým způsobem, jedná se totiž soubor s příponou: „.php“. Obsah tohoto skriptu zpravidla začíná: „`<?php`“ a může i nemusí končit: „`?>`“. Uvnitř se může vyskytovat libovolné množství příkazů. Mohou to být deklarace proměnných, definice cyklů, převzetí proměnné z adresy URL pomocí metody: „`get`“ nebo „`post`“, vypisování textových řetězců, definování funkcí a další. Následuje ukázka krátkého PHP skriptu:

```
<?php
      $mycounter = 1;
      $mystring = 'Hello';
      $myarray = array('One', 'Two', 'Three');

      echo 'Hello';
      echo $mystring;
      echo 3 + 2;
?>
```

[16]

1.2.1.4 Výhody jazyka PHP

Tento jazyk je velice rozšířen. Je podporován velikým množstvím webových hostingů a poskytovatelů internetových služeb. Už jen z tohoto důvodu je dobré znát bližší informace o PHP. Dále je tento jazyk multiplatformní. To znamená, že nezáleží na operačním systému zařízení, na kterém je nainstalován. Proto se uživatel může s PHP setkat na OS Windows, Linuxu, Mac OS X a jiných. Ještě by z definice vyplývalo, že pokud je PHP webová stránka vytvořená na jedné platformě, tak stačí pouze minimální úprava jejího kódu k migraci na platformu jinou. Dále existuje mnoho bezplatných softwarů pro práci s tímto jazykem a volně dostupných projektů, které jsou vytvořeny za pomoci PHP. [9]

1.2.1.5 Nevýhody jazyka PHP

Jazyk sám o sobě je sice použitelný, ale vývojáři ho v dnešní době už málokdy použijí. Z tohoto důvodu jsou vytvořeny frameworky pro práci s PHP, které zahrnují předem definované knihovny. Známé frameworky pro práci s PHP: Nette, Zend, Yii a další. V další řadě se uživatel může potýkat se slabším výkonem aplikace. Tento jazyk jakmile dokončí požadavek, tak neudržuje kontext dané aplikace. Celý kontext následně vytvoří znovu a to aplikaci zatěžuje. [9]

1.2.2 Java

1.2.2.1 Úvodem o jazyku Java

Jedná se o programovací jazyk vyvinutý firmou Sun Microsystems a jeho hlavní předností je, že se jedná o objektově orientovaný nástroj pro tvorbu aplikací. Může se použít k vývoji aplikací, které fungují na bázi desktopové či poskytují nějaké služby prostřednictvím webového serveru. [21] Tato práce se bude zaobírat především metodami sloužícími k výpisu dat z aplikací a jejich následnému zapouzdření do daného formátu.

Java Development Kit (JDK) je prostředek, který obsahuje základní nástroje sloužící k vývoji aplikací platformy Java. Tyto nástroje byly vytvořeny firmou Oracle Corporation v roce 1996 jako verze JDK 1.0 (Java 1.0). Tato nejstarší verze zahrnovala procesní vlákna, zámky a dokonce paměťový modul, avšak postrádala hodně prvků podporující spolehlivost. Postupem času byla vydána verze Java 5, která už měla schopnost využít svou sílu z hlediska procesů a uměla vytvářet bloky. To se dá přirovnat metodě vláken zvanou: „thread pools“, pod kterou se dají představit skupiny nezávislých bloků (pools), ve kterých jsou zapouzdřeny jednotlivá vlákna (threads). V roce 2011 byla vydána další verze: Java 7, která byla navržena velice efektivním způsobem. Procesní vlákna zde měly k dispozici vlastní systém paralelismu (the fork/join Framework), který sice byl mnohem více využitelný z praktického hlediska, ale stále se jednalo o velice komplikovanou komponentu pro vývoj aplikací. Nakonec byla v roce 2014 vyhotovena Java 8, která byla hodně zaměřena na co nejjednodušší kompilaci a návrh zdrojového kódu a na využití více jader u vícejádrových procesorů. Tyto oblasti měly být realizovány pomocí tří prostředků (metod), které byly následující: The Streams API, techniky pro předávání kódu do metod a definice výchozích metod v příslušném interface. Java 8 nabízí nové API (Application Programming Interface) zvané: „Streams“. Tento prvek slouží jako rozhraní (prostředek) pro programování aplikací. [21]

1.2.2.2 Ukázka jazyka Java

Následující ukázka kódu zahrnuje vytvoření veřejné statické metody, která má za úkol (při jejím volání) vytvořit seznam jablek, které jsou těžké (mají více než 150 g). Jelikož se jedná o objektově orientovaný jazyk, tak jablko zde představuje třídu, která se skládá z: atributů, metod, konstruktoru, a dalších věcí. Při zavolání této třídy nějakou metodou se vytvoří její instance (objekt), která už bude nabývat konkrétních (reálných) hodnot. Ukázka vypadá následovně:

```
public static List<Apple> filterHeavyApples(List<Apples> inventory) {
      List<Apple> result = new ArrayList<>();
      for (Apple apple: inventory) {
            if (apple.getWeight() > 150) {
                  result.add(apple);
            }
      }
}
```

[21]

1.2.2.3 Výhody Javy

Celý tento objektově orientovaný jazyk je navržen dynamicky, to znamená, že vývojář má možnost libovolně rozšiřovat knihovnu, která je součástí platformy. Můžou do ní být přidány nové funkce a třídy, vlastní programy a další nástroje. Dále je Java určena jak pro malé, tak pro velké aplikace. Díky již zmíněnému mechanismu řízení procesních vláken (u verze Java 8) dokáže obstát při rozsáhlejších zpracovávání dat. Také nabízí celou řadu nástrojů pro testování aplikací s menším rozsahem. Velkým pozitivem se stává možnost úpravy aplikace koncovým uživatelem. Nedostatky nebo vylepšení si může uživatel doladit podle vlastního úsudku. V poslední řadě tento jazyk v oblasti vytváření skriptů může uživateli nabízet DSL (Domain-specific language). Tato služba slouží jako popis programovacího jazyka, který je zaměřen (specializován) na práci s konkrétní problémovou doménou. Ze základního hlediska je jazyk Java navržen obecně pro vytváření aplikací problémových domén různého charakteru, proto se jazyk skrze abstrakci a vhodný výrazový slovník omezí pouze pro potřeby dané problémové domény. [18]

1.2.2.4 Nevýhody Javy

Přes řadu výhod se vývojáři mohou potýkat i s nevýhodami tohoto produktu. Např. se může jednat o samotný zdrojový kód, který je sice velice podrobný, popsaný a stává se tak pro uživatele velice dobře čitelným, ale tyto rysy mohou mít i negativní dopad na celou aplikaci. Pokud je aplikace značně rozsáhlá a bude obsahovat chybu logického charakteru ve zdrojovém kódu, tak se kód kvůli své délce a uspořádanosti bude špatně ladit. Ve vývojových prostředích je k dispozici debugger, ale ani ten nemusí odhalit chybu. Proto vývojář může strávit hodně dlouhou dobu hledáním cizí nebo i dokonce vlastní chyby. [21]

JVM (Java Virtual Machine) je soubor programových a datových struktur, který díky virtuálnímu stroji (modul) spouští programy a skripty v jazyce Java. Bytecode je druhem kódu, do kterého Java kompiluje zdrojový kód. Je spouštěn právě pomocí JVM. Z hlediska tohoto procesu kompilace je aplikace vyvinutá v Javě několikrát pomalejší, než aplikace sestavěné za pomoci nativního jazyka (C/C++). Rychlost v provedených procesech se sice postupně navyšuje díky mechanismům pro zpracování procesních vláken, ale tento nástroj je stále nedostačující. Pokud by ještě byla řeč o procesních vláknech, tak efektivní využívání více jader procesoru může být další nevýhodou. Spuštění aplikace napsané v Javě může vyvolat mnoho procesních vláken a ty za běhu programu využijí patřičnou část výkonu procesoru. Poté může být procesor přetížen. V poslední řadě aplikace vytvořené v Javě mají značně vyšší paměťovou náročnost, než aplikace konstruovány za pomoci nativního jazyka. [18]

2 Praktická část

2.1 Účel praktické části

Hlavním cílem této sekce je vyhodnocení optimálního výstupního formátu souborů dat pro práci s velkým a malým množstvím. Dále bude navržena aplikace, u které se bude posuzovat metodika generování výstupních souborů za pomoci jazyka Java a PHP. Prototyp této aplikace bude zaměřen na co nejširší možnost generování dat v různých formátech souborů, po vygenerování by data byla připravena pro další použití. Dále by aplikace zahrnovala vlastní vyhodnocovací logiku, která by vložená uživatelská data změřila z hlediska jejich rozsahu a navrhla by optimální výstupní formát pro zpracování. Z větší míry je předpokladem, že se bude jednat o data databázového charakteru, ale toto hledisko by nebylo striktně omezeno. Program by mohl být použitelný i na serverové bázi, neboť oba jazyky tuto možnost připouštějí.

Nejprve se provede hodnocení optimálního exportovaného formátu dat z databáze na základě různých charakteristik. Výsledkem bude nejvíce užitečný formát souborů dat pro práci s malým a velkým množstvím dat, který bude posuzován na základě všeobecné náročnosti, vypovídací schopnosti, rychlosti apod. Toto hodnocení by mělo sloužit jako samotný základ pro logiku doporučení specifického formátu pro daný export dat.

V dalším kroku bude navržena již zmíněná aplikace, jejíž hlavní funkcí by bylo generování různých výstupních formátů souborů dat. Byly vybrány 2 programovací jazyky, pomoci nichž by mohla být daná aplikace sestavena. V této sekci se bude posuzovat, jak efektivně si každý jazyk dokáže poradit se zpracováním velkého a malého množství dat.

Ve finální části bude vyhodnoceno, které formáty souborů dat jsou vhodné pro generování velkého a malého množství dat pro oba zvolené jazyky. Poté bude následovat klasifikace vhodného formátu pro práci s oběma množstvími z obecného hlediska používání a různých charakteristik.

2.2 Metoda vážených kritérií pro exportované formáty souborů dat

2.2.1 Postup při použití metody

Metoda byla zaměřena na zjištění nejvhodnějšího formátu souborů určeného pro exportování dat ze systému MySQL při malém a velkém množství dat obsaženého v databázi. Na základě podkladů z teoretické části (především výhod a nevýhod jednotlivých formátů) a požadavků aplikace specializující se na generování souborů bylo zvoleno celkem 5 vlastností, které formáty nesly, a ke každé z těchto vlastností byla přiřazena váha. Váha označuje koeficient, který klade důraz na podstatnost jednotlivých vlastností. Tento koeficient nabýval hodnot z intervalu 0 až 1, čím vyšší hodnotu vlastnost nabývala, tím se stala podstatnější pro analýzu. Dále byla zvolena hodnotící škála s příslušným intervalem hodnot od 0 do 10 a hodnoty tohoto intervalu byly přiřazovány jednotlivým formátům pro každou z vlastností. Podobně jako u váhy byla vyšší hodnota přínosnější pro daný formát. Hodnoty hodnotící škály byly přiřazovány především na základě ukázkového příkladu exportu tabulky adresa (pro malý obsah dat) a export databáze „suchava1“ (pro velký obsah dat), který byl exportován do všech testovaných formátů souborů a následně posouzen v této metodě dle hledisek vytvořených skriptů.

V konečné fázi byla sestavena tabulka, kde se váhový koeficient násobil s příslušným řádkem každého formátu souboru dat. Výsledky násobení jsou uvedeny v dolním indexu hodnoty hodnotící škály. Pro tyto násobky se provedl součet pro každý formát dat odděleně a v posledním řádku tabulky se tímto způsobem docílilo výsledných hodnot. Všechny kroky výpočtů jsou zahrnuty v přílohách této práce.

Metoda byla aplikována pro 2 skupiny dat. Pro malou skupinu obsahu dat, která představovala databázi o několika tabulkách (popř. ukázková tabulka Adresa), které obsahovaly několik řádků. Velká skupina dat pokrývala širší oblast databáze a byla tvořena řádově více než jedním tisícem řádků.

2.2.2 Popis testovaných dat

Na základě analýzy jednotlivých vlastností formátů dat byly vytvořeny 2 databáze. První z nich obsahovala malý obsah dat, jednalo se totiž o miniaturní databázi, která sloužila jako ukázka exportů v teoretické části této práce. Byla v ní obsažena 1 tabulka, která zahrnovala několik řádků dat. Resp. byla exportována pouze tabulka, nikoliv celá databáze. Pokud bude nahlédnuto na obsah druhé databáze s větším množství dat, tak se bude jednat o stejný základ jako databáze v prvním případě, akorát data v ní obsažena jsou rozšířena. V podstatě bylo vytvořeno několik nových tabulek s danými daty a jedna z těchto tabulek byla naplněna více než jedním tisícem řádků dat.

2.2.3 Měření rychlosti exportu souborů

2.2.3.1 Postup při měření

Při měření doby vyexportování jednotlivých souborů se použilo nástroje phpmyadmin na serveru edu.uhk.cz. Zde se exportovaly již zmíněné databáze (předchozí kapitola) malého a velkého obnosu dat. K měření doby exportu se používaly implementované nástroje internetového prohlížeče Google Chrome. Všechny hodnoty byly naměřeny za totožných podmínek.

2.2.3.2 Naměřené hodnoty pro malé množství dat

Formát Souboru:	XML	PDF	CSV	SQL	MS Word	CSV pro MS Excel
Rychlost vygenerování (ms):	174	397	202	199	185	169

Tabulka 4: Rychlost exportu souborů z MySQL při malém množství dat [vlastní zpracování]

2.2.3.3 Naměřené hodnoty pro velké množství dat

Formát Souboru:	XML	PDF	CSV	SQL	MS Word	CSV pro MS Excel
Rychlost vygenerování (ms):	285	2270	312	423	267	375

Tabulka 5: Rychlost exportu souborů z MySQL při velkém množství dat [vlastní zpracování]

2.2.4 Postup při aplikování metody pro malý obsah dat

Jak již bylo řečeno, bylo určeno 5 vlastností, které byly pozorovány a následně hodnoceny pomocí metody vážených kritérií. Těmto vlastnostem byla stanovena určitá váha, která sloužila jako koeficient při stanovení důrazu jednotlivých vlastností. Jelikož se jedná o malý obsah dat, tak je předpokládáno, že pro uživatele bude největší prioritou rychlost vyexportování dat z databáze. Z tohoto důvodu byla této vlastnosti přidělena nejvyšší hodnota z intervalu váhy (1). Dále se analyzovala vlastnost: „náročnost pro uživatele“. Vzhledem k obnosu dat se přiřadila jedna z vyšších hodnot (0,9), neboť by uživatel při exportování těchto dat nemusel mít potřebné znalosti

o jednotlivých formátech dat a nemusel by vědět, který použít nebo který z nich je optimální pro specifická data. Následovala vypovídací schopnost, která zmiňuje rozsah metadat jednotlivých souborů. Její důležitost spočívá v informacích, které nese, neboť není známa činnost, pro kterou budou data po exportu určena. Mohou se nechat vytisknout, spravovat, editovat, přeposílat a využít i více způsoby. Proto byla přidělena hodnota z vyšších řad (0,8). Zbývající 2 vlastnosti spolu blízce souvisí. Jedná se o délku obsahu souboru a velikost souboru. Proto z hlediska vah byly vyhodnoceny velice podobným způsobem (0,5 a 0,4). Umístily se až za ostatními vlastnostmi opět z důvodu nízkého obsahu dat jednotlivých souborů, neboť dá se tvrdit, že velikosti těchto malých souborů jsou zanedbatelné.

Nyní bude doloženo, jakým způsobem byly přiřazeny hodnoty z hodnotící škály jednotlivým formátům souborů dat. V první řadě byly hodnoty přiřazeny k vlastnosti: „rychlost exportu“. Tato data byla zpozorována na základě časových údajů, které byly měřeny v předchozí kapitole. Hodnoty byly vypočteny následovným způsobem: jednotlivé naměřené hodnoty byly seřazeny podle velikosti, hodnotit se začalo od nejrychlejšího vyexportovaného souboru, který nabyl nejvyšší hodnoty (10). Dále se bral v úvahu rozdíl mezi sousedními časy (resp. velikost tohoto úseku) a na tomto základě se vypočetly jednotlivé poměry.

Vlastnost „náročnost pro uživatele“ byla posuzována především podle samotných dat, která by údajně měl uživatel nalézt v exportovaném souboru. Slovo „náročnost“ napovídá, že byla zohledněna celková představa složitosti. Např. soubor CSV zahrnoval minimální obsah dat, a proto se data zdála velice dobře pochopitelná z hlediska informace, kterou nesly. CSV tedy nabylo nejvyššího hodnocení (10). Jelikož data pro formát MS Excel byla téměř totožná (CSV určené pro MS Excel - použitelné v tabulkovém editoru), tak proto zde bylo hodnocení stejné (10). PDF dosáhlo také velice dobrých výsledků (9), neboť obsah byl malý a navíc zaobalen pomocí vykreslené tabulky (názorná ukázka se nachází v přílohách této práce). Na druhé straně, nízké hodnoty (4) byly zaznamenány u formátu SQL, neboť uživatel musí mít alespoň základní znalosti o jazyce SQL a to jak z hlediska DML, tak i DDL (viz teoretická část). Dále negativně posouzeno (3) bylo XML, neboť tu také byly potřebné nějaké vědomosti o stromové struktuře, při otevření tohoto souboru se data nepřehledně roztáhla na jeden řádek, a jak již bylo nastíněno v teoretické části, jsou zapotřebí vědomosti ohledně problematiky DTD.

U vypovídací schopnosti se dbalo na analyzování jednotlivých souborů, kde se brala v úvahu všechna data, která nesla nějakou informaci navíc oproti exportovaným datům. Analýza probíhala pomocí volně dostupného nástroje zvaného: Metadata Extractor (dostupný na: https://github.com/drewnoakes/metadata-extractor), který slouží pro extrahování metadat z daných souborů. Tento nástroj byl použitelný pro formáty: MS Excel, MS Word, XML a PDF. Ukázky extrakce jsou v k dispozici v přílohách této práce. Všechny extrakce byly provedeny pro malý obsah dat, jednalo se o některé soubory, které byly vyexportovány z databáze nebo vygenerovány pomocí jednoho z programovacích jazyků. Při analýze CSV se použilo jedné z ukázek, která nebyla vytvořena v rámci této práce (posloužila pouze pro hodnocení daných metadat). U formátu SQL se využilo jedné z ukázek, která byla vyexportována z databáze.

Z hlediska porovnání jednotlivých formátů (resp. obsažených metadat) bylo CSV vyhodnoceno nejhůře. Analýza byla soustředěna na následující ukázku, která pojednává o metadatech formátu CSV, která jsou znázorněna pomocí XML:

```
<?xml version="1.0" encoding="UTF-8"?>

<meta xmlns="http://mafr.de/ns/meta-0.1#">
  <file-location format="csv" compression="gzip" separator="\t">
    <chunk location="data_1.csv.gz" size="123456"/>
    <chunk location="data_2.csv.gz" size="34354"/>
    <!-- more chunk declarations ... -->
    <chunk location="data_N.csv.gz" size="7890123"/>
  </file-location>

  <data-fields>
    <string-field name="USER_ID"/>
    <string-field name="NAME"/>
    <continuous-field name="AGE" missing="0"/>
    <categorical-field name="COUNTRY" missing="?">
      <category value="DE"/>
      <category value="UK"/>
      <category value="OTHERS"/>
    </categorical-field>
  </data-fields>
</meta>
```

Obrázek 2: Ukázka metadat souboru CSV zapsaných pomocí XML [10]

Je možné si povšimnout, že na ukázce je znázorněn oddělovač (`separator`) s příslušným zadaným znakem (`\t`), který je u tohoto formátu velice důležitý (více o struktuře CSV je obsaženo v teoretické části této práce). Dají se vypozorovat některá textová pole (`string-field`) a několik informací o složce a její lokaci (`file-location`). Jsou zde postrádána data jako: autor souboru, datum (i čas) vytvoření nebo změny, software nebo operační systém (se kterým soubor přišel do styku), použitá znaková sada dat, apod. Vzhledem k minimálnímu obsahu těchto dat a vzhledem k obsahu metadat ostatních souborů bylo CSV vyhodnoceno na poslední pozici (1) a jelikož formát MS Excel byl exportován na základě CSV, tak byl vyhodnocen stejným způsobem (1). Pokud bylo nahlédnuto na data ve formátu MS Word (ukázka viz přílohy), tak bylo zpozorováno mnoho údajů ohledně velikosti souboru, počet stran dokumentu, časové údaje o manipulaci s dokumentem, informace o autorech a další. Vyexportovaná data byla rozčleněna do jednotlivých bloků pro DML a DDL, které byly popsány příslušnými nadpisy, a v blocích se nacházely jednotlivé hodnoty se záhlavím uspořádané do tvaru tabulky pomocí tabulátorů. Avšak mřížky tabulky sházely a skutečná tabulka nebyla vygenerována. Na základě analýzy bylo tomuto formátu uděleno 7 bodů. Při průzkumu metadat souboru PDF (ukázka viz přílohy) bylo zjištěno, že tento formát si vedl podobným způsobem jako MS Word. Rozdílem byly údaje o rozměrech příslušných stran, o funkcích zabezpečení a z hlediska vyexportovaných dat byla data ucelena do tabulky. PDF získalo proto lepší hodnocení (8). Při hodnocení formátu SQL bylo využito ukázky exportu při malém množství dat z databáze (viz přílohy), protože samotný soubor metadata již zahrnoval. Data byla striktně databázového charakteru a nesly informace o serveru (ze kterého byla data exportována), mnoho údajů o verzích, informace o znakové sadě (podrobnější, než u předchozích formátů), atd. Z pohledu samotných dat, vyexportovaná data byla vhodně rozčleněna na sekci DML a DDL, samotný jazyk SQL již v základu nesl informace o datových typech jednotlivých dat, o výchozí hodnotě při absenci údajů apod. Vzhledem k těmto poznatkům bylo SQL ohodnoceno 9 body. Poslední zkoumaný formát XML (ukázka viz přílohy) byl z hlediska obsahu metadat velice podobný jako předchozí zkoumaný formát. Rozdílem by mohlo být zaobalení dat do párových tagů. Vzhledem k podobné povaze bylo hodnocení rozšířeno i o teoretická fakta vypovídajících o těchto formátech. Při zvážení exportu dat jiného charakteru, než databázového, by si XML vedlo lépe. O flexibilitě tohoto formátu hovoří Castro [1] ve svých studiích, kde zmiňuje mnoho oblastí (možné pro všechny

oblasti), kde se dá XML využít. O SQL se zmínili Connolly, Thomas a Carolyn [2], v jejich tvorbě je zdůrazněno, že se jazyk používá především pro vytváření databází a jejich struktur. Proto bylo XML ohodnoceno o jednotku lépe (10), než SQL.

Délka obsahu souboru byla hodnocena podle velikosti dat, která se nacházela v exportovaném souboru. Data v CSV a v MS Word byla v rozsahu 3 řádků a u XML a SQL data zabírala výraznější množství obsahu. Podle velikosti zabraného prostoru uvnitř souboru byly přiřazeny jednotlivé hodnoty. Pro lepší představu můžou posloužit přílohy, kde se nacházejí všechny vyexportované soubory.

Vlastnost: „velikost souboru“ byla posuzována na základě vyexportovaných souborů z ukázkové databáze „suchava1“, kde pro malý obsah dat byla exportována pouze 1 tabulka obsahující 3 řádky dat. Soubory jsou zahrnuty v přílohách této práce. Jednotlivé velikosti souborů byly následující:

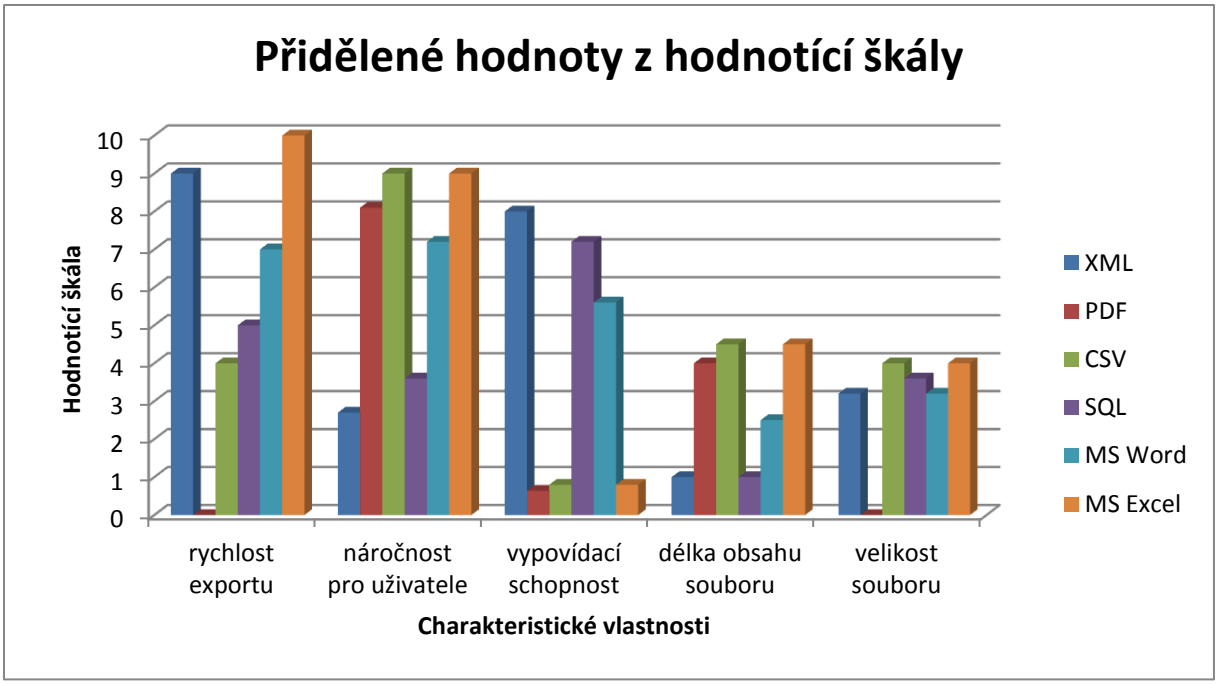
Formát Souboru:	XML	PDF	CSV	SQL	MS Word	CSV pro MS Excel
Velikost (kB):	3	179	1	2	3	1

Tabulka 6: Velikosti vyexportovaných souborů z MySQL při malém obsahu dat [vlastní zpracování]

2.2.5 Aplikace metody pro malý obsah dat

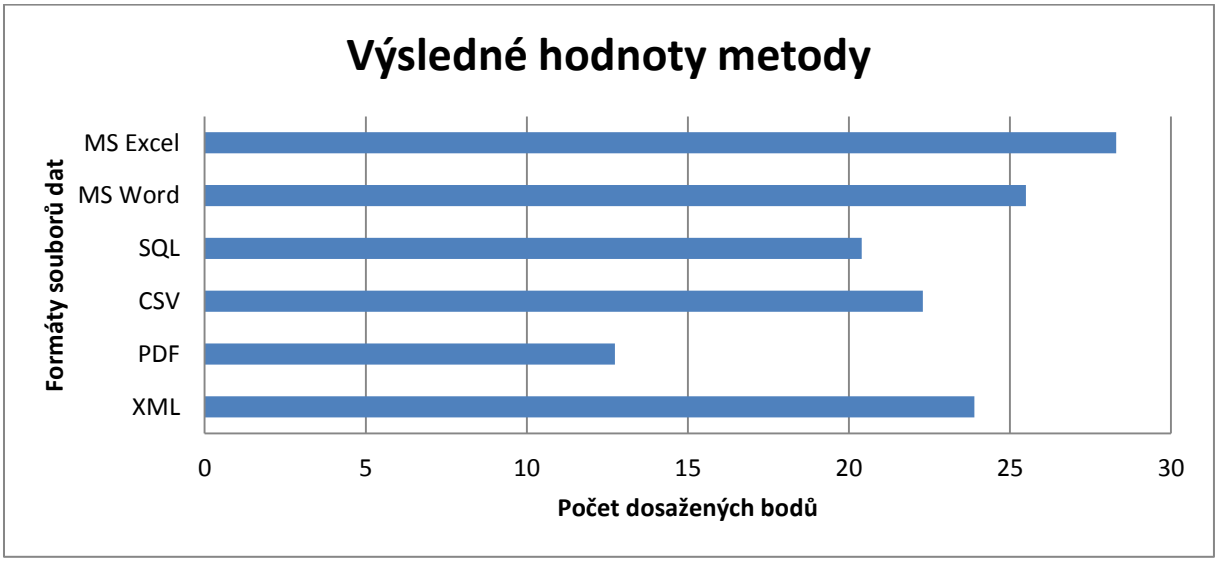
váha	vlastnosti\formáty souborů	XML	PDF	CSV	SQL	MS Word	MS Excel
1	rychlost exportu	$9_{(9)}$	$0_{(0)}$	$4_{(4)}$	$5_{(5)}$	$7_{(7)}$	$10_{(10)}$
0,9	náročnost pro uživatele	$3_{(2,7)}$	$9_{(8,1)}$	$10_{(9)}$	$4_{(3,6)}$	$8_{(7,2)}$	$10_{(9)}$
0,8	vypovídací schopnost	$10_{(8)}$	$8_{(0,64)}$	$1_{(0,8)}$	$9_{(7,2)}$	$7_{(5,6)}$	$1_{(0,8)}$
0,5	délka obsahu souboru	$2_{(1)}$	$8_{(4)}$	$9_{(4,5)}$	$2_{(1)}$	$5_{(2,5)}$	$9_{(4,5)}$
0,4	velikost souboru	$8_{(3,2)}$	$0_{(0)}$	$10_{(4)}$	$9_{(3,6)}$	$8_{(3,2)}$	$10_{(4)}$
		23,9	12,74	22,3	20,4	25,5	28,3

Tabulka 7: Metoda vážených kritérií pro exportované formáty při malém obsahu dat [vlastní zpracování]



Graf 1: Metoda vážených kritérií pro exportované formáty při malém množství dat [vlastní zpracování]

2.2.6 Výsledky metody pro malý obsah dat



Graf 2: Výsledky metody vážených kritérií pro exportované formáty při malém množství dat [vlastní zpracování]

2.2.7 Postup při aplikování metody pro velký obsah dat

V této sekci bylo znovu zkoumáno celkem 5 charakteristických vlastností příslušných formátů souborů dat. Oproti malému množství dat byla pozměněna váha těchto vlastností, neboť při exportování většího obsahu dat se může na data pohlížet rozdílným způsobem. Jelikož se bude jednat o velké množství dat, tak se prioritní vlastností stala velikost souboru (0,9). Předpokládá se, že pro uživatele bude rychlost exportování stále podstatná, ale vzhledem k ostatním vlastnostem se nepatrně změnil váhový koeficient (0,8). Je zřejmé, že rozsah jednotlivých metadat zůstane při změně

velikosti dat zachován. Váha nabyla sice jiné hodnoty (0,5), ale pozice této vlastnosti zůstala nepozměněna. Jelikož od uživatele, který bude manipulovat s vysokým obnosem dat, se očekává, že již bude mít nějaké zkušenosti a potřebné znalosti, tak náročnost pro uživatele nabyla nižší hodnoty váhy (0,3). Vzhledem k poslední vlastnosti, očekává se, že z důvodu vysokého obnosu dat velikost souboru bude natolik prioritní, že nebude mít význam zkoumat délku obsahu. Proto vlastnost: „délka obsahu souboru“ se stala téměř bezvýznamnou (0,1).

Nyní bude doloženo, jakým způsobem byly přiřazeny hodnoty z hodnotící škály jednotlivým formátům. Vlastnost: „velikost souboru“ byla posuzována na základě poměru velikostí vyexportovaných souborů z ukázkové databáze „suchava1“, kde byla exportována celá databáze obsahující přes tisíc řádků dat. Soubory jsou zahrnuty v přílohách této práce. Jednotlivé velikosti souborů byly následující:

Formát Souboru:	XML	PDF	CSV	SQL	MS Word	CSV pro MS Excel
Velikost (kB):	224	170	29	44	155	29

Tabulka 8: Velikosti vyexportovaných souborů z MySQL při velkém obsahu dat [vlastní zpracování]

U rychlosti exportování byla data zpozorována na základě časových údajů, které byly měřeny v jedné z předchozích kapitol. Hodnoty byly vypočteny následovným způsobem: jednotlivé naměřené hodnoty byly seřazeny podle velikosti, hodnotit se začalo od nejrychlejšího vyexportovaného souboru, který nabyl nejvyšší hodnoty (10). Dále se bral v úvahu rozdíl mezi sousedními časy (resp. velikost tohoto úseku) a na tomto základě se vypočetly jednotlivé poměry.

Z hlediska hodnocení vypovídací schopnosti se dbalo na posuzování malého množství dat, proto se hodnoty shodovaly v různých obnosech dat (metadata byla totožná), rozdíl spočíval pouze v použití váhového koeficientu. Ukázky metadat jsou k dispozici v přílohách této práce.

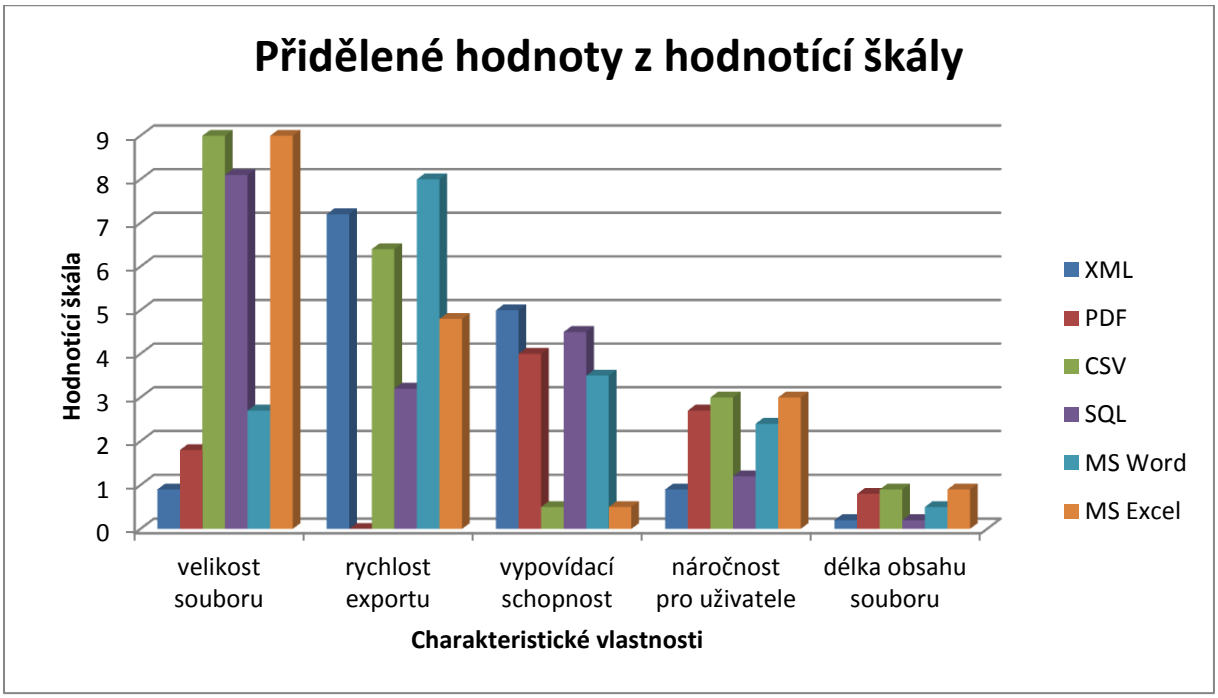
Vlastnost „náročnost pro uživatele“ byla opět analyzována na základě dat z předchozího průzkumu malého obnosu dat. V úvahu se brala celková představa složitosti a náročnosti pro uživatele, která byla hodnocena z vyexportovaných souborů (ukázky viz přílohy). Jelikož toto hledisko bylo poměrově srovnatelné, tak se použilo hodnot, které byly vyhodnoceny u malého množství dat. Změněn byl pouze váhový koeficient.

U hodnocení poslední vlastnosti (délka obsahu souboru) se opět hledělo na poměrové rozdíly u malého a velkého množství dat. Rozdílem se stal pouze váhový koeficient a data byla shodná pro oba obnosy dat.

2.2.8 Aplikace metody pro velký obsah dat

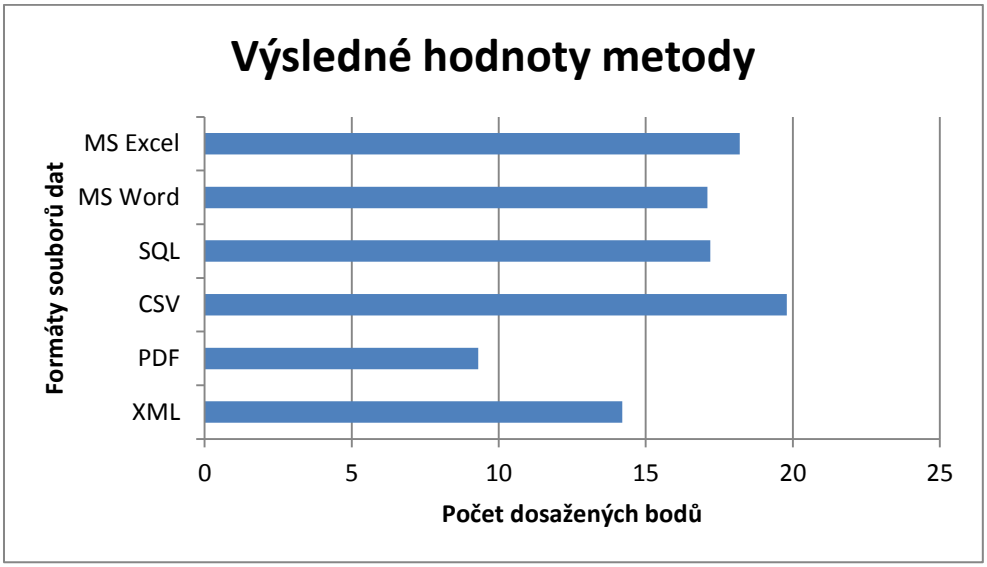
váha	vlastnosti\formáty souborů	XML	PDF	CSV	SQL	MS Word	MS Excel
0,9	velikost souboru	$1_{(0,9)}$	$2_{(1,8)}$	$10_{(9)}$	$9_{(8,1)}$	$3_{(2,7)}$	$10_{(9)}$
0,8	rychlost exportu	$9_{(7,2)}$	$0_{(0)}$	$8_{(6,4)}$	$4_{(3,2)}$	$10_{(8)}$	$6_{(4,8)}$
0,5	vypovídací schopnost	$10_{(5)}$	$8_{(4)}$	$1_{(0,5)}$	$9_{(4,5)}$	$7_{(3,5)}$	$1_{(0,5)}$
0,3	náročnost pro uživatele	$3_{(0,9)}$	$9_{(2,7)}$	$10_{(3)}$	$4_{(1,2)}$	$8_{(2,4)}$	$10_{(3)}$
0,1	délka obsahu souboru	$2_{(0,2)}$	$8_{(0,8)}$	$9_{(0,9)}$	$2_{(0,2)}$	$5_{(0,5)}$	$9_{(0,9)}$
		14,2	9,3	19,8	17,2	17,1	18,2

Tabulka 9: Metoda vážených kritérií pro exportované formáty při velkém obsahu dat [vlastní zpracování]



Graf 3: Metoda vážených kritérií pro exportované formáty při velkém množství dat [vlastní zpracování]

2.2.9 Výsledky metody pro velký obsah dat



Graf 4: Výsledky metody vážených kritérií pro exportované formáty při velkém množství dat [vlastní zpracování]

2.3 Metoda vážených kritérií pro export dat ve dvou různých programovacích jazycích

2.3.1 Popis testovaných dat

Na základě pozorování bylo pro každý jazyk vytvořeno 10 miniaplikací, které měly za úkol vygenerovat příslušné formáty souborů dat. Tyto miniaturní spustitelné programy byly následně testovány za stejných podmínek hlavně z hlediska chování a rychlosti generování samotných formátů. Oba jazyky byly hodnoceny odděleně, neboť se na jedné straně pojednávalo o skriptovacím jazyce (PHP) a na straně druhé o malých desktopových aplikacích, které byly sestaveny na základě objektově orientovaného jazyka Java.

Pro každý formát bylo posuzováno chování daného jazyka pro velké a malé množství dat odděleným způsobem. Pomocí jazyka PHP bylo vytvořeno 5 skriptů, které při spuštění vygenerovaly jejich příslušný soubor o daném formátu. Každý z těchto skriptů měl implementovánu externí knihovnu, která obsahovala metody vhodné ke generování. Knihovny byly různého charakteru, neboť jich existuje veliká škála a i pouhá dostupnost těchto knihoven byla následně hodnocena v metodě vážených kritérií. Jednalo na např. o knihovny typu „Response“, u kterých bylo zapotřebí vytvořit instanci dané třídy, kde se v konstruktoru uváděla data určená ke generování a název výstupního souboru. Dále se zavolala jednoduchá metoda, která generovaný soubor nabídla uživateli ke stažení. Tyto knihovny byly použity při generování dat v jazyce PHP ve formátech CSV a MS Word. U zbylých formátů byl použit balík „PHPOffice“, který obsahoval knihovny pro generování všech formátů dat pro aplikace obsažené v kancelářském balíku Microsoft Office. Práce s těmito knihovnami byla již složitější a uživatel už musel nahlédnout do příslušné dokumentace. Všechny tyto knihovny byly volně k dispozici prostřednictvím sítě Internet. U některých skriptů bylo využito frameworku Nette, aby bylo zjištěno, jak efektivně dokáže zvolený framework s knihovnami pracovat. Po sestavení aplikací byly aplikace testovány a malé množství dat představovalo několik řádků výstupu, řádově desítky. U velkého množství dat byly skripty rozšířeny většinou tak, aby obsahovaly přes jeden tisíc výstupních dat a taktéž bylo pozorováno jejich vygenerování.

Při uchopení jazyka Java bylo obdobně vytvořeno celkem 10 miniaplikací pro velké a malé množství dat. Pro generování CSV formátu byla využita již vestavěná knihovna v Netbeans IDE 8.0.1, jejíž hlavní třída se nazývala: „FileWriter“,

která dědila metody od „OutputStreamWriter“. Dále se využívalo externě přidaných knihoven, především od organizace Apache Software Foundation (ASF). Jednalo se např. o Java API, které bylo vytvořeno speciálně pro generování dokumentů od společnosti Microsoft.

U testovaných dat byly zvoleny celkem 4 charakteristické vlastnosti, které vypovídaly informace o základních rysech generovaných souborů. V prvé řadě byla vybrána vlastnost rychlost generování určitého souboru, neboť měření v této situaci bylo vhodné a vypovědělo mnoho informací o schopnostech obou testovaných jazyků. Postup, který byl použit pro tuto vlastnost, a jeho výsledky jsou uvedeny v sekci, která se nachází níže. Dále se zaznamenával možný výskyt chyb při generování jednotlivých souborů. Tato vlastnost byla zkoumána, neboť podle poznatků z teoretické části práce se jevila jako velice důležitá. Vlastnost: „podpora knihoven“ byla testována hlavně především z důvodu předpokladu, který vypovídá o navržené aplikaci. Aplikace by byla naprogramována a sestavena pomocí některého ze zkoumaných jazyků a rozhodně by byla nezbytně nutná implementace knihoven. Proto další vybraná vlastnost se týkala charakteristiky knihoven. V poslední řadě se bralo v úvahu, že programátor, který má danou aplikaci sestavit, by musel mít určité znalosti (i možné zkušenosti) s daným jazykem a příslušnými knihovnami. Z tohoto důvodu byla analyzována poslední vlastnost: „složitost jazyka“.

2.3.2 Měření rychlosti vygenerování souborů

2.3.2.1 Postup při měření

Pro každý jazyk bylo vytvořeno celkem 10 skriptů, které byly následně testovány. Pro jazyk PHP se analyzovaly příslušné skripty pomocí localhostového serveru EasyPHP, kde přes internetový prohlížeč byly spuštěné skripty, které obsahovaly hlavní metodu pro exportování dat. Po spuštění dané skripty vygenerovaly příslušný obsah dat do daného formátu a nabídly vytvořený soubor ke stažení. V tomto okamžiku se měřila doba vygenerování příslušného skriptu pomocí nástrojů zvoleného prohlížeče Google Chrome, kde byla zachycována doba mezi zahájením hlavní metody daného skriptu a okamžikem nabídnutí souboru ke stažení.

V případě jazyka Java byly vytvořeny miniaturní spustitelné aplikace, které obdobným způsobem měly jako hlavní funkci vygenerovat příslušný soubor s daným formátem dat. Měření probíhalo ve vývojovém prostředí NetBeans IDE verze 8.0.1, kde se pomocí implementovaných nástrojů analyzovala doba od počátku spuštění aplikace do jejího sestavení.

2.3.2.2 Naměřené hodnoty

Formát souboru:	XML	PDF	CSV	MS Word	MS Excel
Rychlost vygenerování (ms):	12	917	340	494	417

Tabulka 10: Rychlost generování souborů v jazyce PHP při malém množství dat [vlastní zpracování]

Formát Souboru:	XML	PDF	CSV	MS Word	MS Excel
Rychlost vygenerování (ms):	21	18660	350	1430	226

Tabulka 11: Rychlost generování souborů v jazyce PHP při velkém množství dat [vlastní zpracování]

Formát souboru:	XML	PDF	CSV	MS Word	MS Excel
Rychlost vygenerování (ms):	19	495	112	358	211

Tabulka 12: Rychlost generování souborů v jazyce Java při malém množství dat [vlastní zpracování]

Formát souboru:	XML	PDF	CSV	MS Word	MS Excel
Rychlost vygenerování (ms):	128	5600	233	1132	1269

Tabulka 13: Rychlost generování souborů v jazyce Java při velkém množství dat [vlastní zpracování]

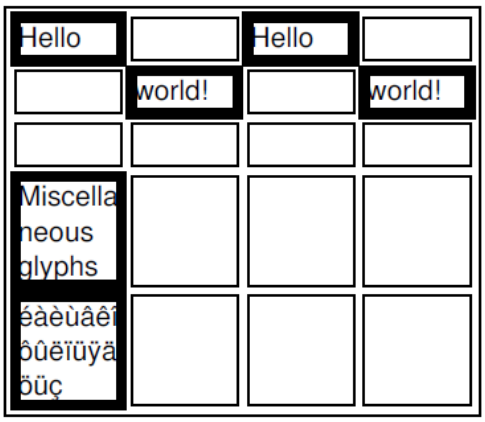
2.3.3 Metoda vážených kritérií pro export dat v jazyce PHP do různých formátů souborů

2.3.3.1 Postup při aplikování metody pro malý obsah dat

Jak již bylo zmíněno, byly zvoleny 4 charakteristické vlastnosti, které byly zkoumány. Důvod výběru právě těchto vlastností je uveden výše. Ke každé z těchto vlastností byl přiřazen koeficient váhy, který určoval důraz kladený na určitou vlastnost. Jelikož se jednalo o generování malého množství dat, tak nejvyšší hodnota byla u rychlosti vygenerování samotných souborů. Pokud by uživatel chtěl vygenerovat pouze malou množinu dat, tak se předpokládá, že by tato vlastnost měla nejvyšší prioritu. Hodnota koeficientu vlastnosti: „chybovost při generování“ byla vyhodnocena jako nižší, neboť čím méně dat se bere v úvahu, tak se očekává, že tím méně chyb budou tato data obsahovat. Dále se posuzovala podpora knihoven. Váha zde nabývala nižší hodnoty z důvodu priority dvou předchozích vlastností. Jako poslední byla testována složitost jazyka, jejíž koeficient byl posouzen jako nejnižší opět na základě priority.

Nyní bude doloženo, jakým způsobem byly přiřazeny hodnoty z hodnotící škály jednotlivým formátům souborů dat. Při hodnocení vlastnosti: „rychlost vygenerování“ byly použity údaje z měření rychlosti generování souborů v jazyce PHP při malém množství dat, které jsou uvedeny výše. Byl vytvořen poměr naměřených dat v milisekundách a ten byl následně vepsán do tabulky.

Při hodnocení chybovost při generování se postupovalo následujícím způsobem. Vygenerování samotných souborů občas nebylo zcela korektní a ve výstupním souboru se nalézaly chyby. Např. v souboru o formátu PDF se nalézaly nevhodně veliké tabulky, které skrývaly delší obsah jednotlivých buněk. Proto tento formát nabyl nižší hodnoty z hodnotící škály, než formáty ostatní. Následuje ukázka vygenerované tabulky v PDF:



Obrázek 3: Vygenerovaná tabulka v PDF za pomoci jazyka PHP při malém množství dat [vlastní zpracování]

MS Word a MS Excel měly občas problémy s rozpoznáním dané znakové sady, tento fakt se projevil při použití diakritiky generovaných dat. U zbylých formátů nebyly nalezeny chyby, proto se hodnotilo na základě poznatků z teoretické části této práce. Proto byl formát CSV vyhodnocen jako nejméně chybový, protože při generování malých dat obsahoval minimální množinu prvků. Hned na druhém místě byl formát XML, který již zmíněných prvků obsahoval více.

Z hlediska vlastnosti: „podpora knihoven“ se hodnotilo na základě průzkumu všech možných dostupných knihoven, které sloužily pro generování daných formátů. Také v hodnocení byla posouzena složitost použití těchto knihoven podle vlastních poznatků při vytváření skriptů. Nejlepší hodnocení měl formát CSV, protože knihovna pro generování byla již většinou součástí vývojových prostředí. Veliká škála knihoven pro generování formátu XML byla volně dostupná prostřednictvím sítě Internet. Navíc tyto knihovny pro generování XML podporovaly vytváření takové struktury tohoto formátu, která byla dále spustitelná v programu Microsoft Excel. Dále pro oba formáty vytvořené společností Microsoft byla k nalezení menší množina knihoven, než u formátů ostatních. Např. zde byla nalezena již zmíněná knihovna „PHPWord“, která spadala do balíku knihoven „PHPOffice“. Tato knihovna měla k dispozici mnoho podob a modifikací. Na posledním místě se umístilo PDF. Knihoven bylo k nalezení pouze několik (knihovna mPDF, PDF Response, apod.) a navíc se s těmito knihovnami nepracovalo snadno. Jejich dokumentace byla značně rozsáhlá a mnohokrát zkoumaná knihovna vypisovala chybové reporty. Proto PDF u této vlastnosti nabylo nejnižší hodnoty.

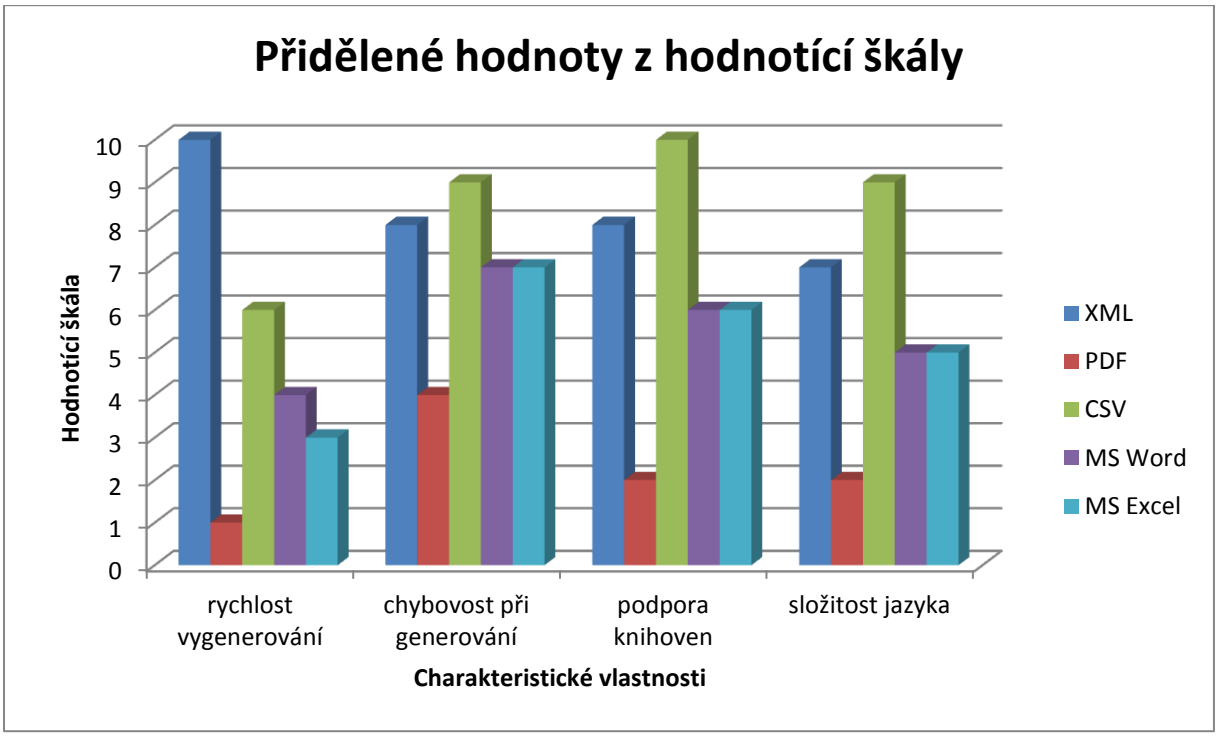
Jako poslední se posuzovala složitost jazyka. Zde se analyzovala celková přehlednost kódu daných skriptů, odhadnutelnost používaných metod a způsob, kterým jazyk PHP dokáže uchopit generování příslušného souboru s daným formátem dat. Ukázky hlavních metod pro generování jednotlivých formátů jsou zahrnuty v přílohách této práce. Nejjednodušší se jevily skripty pro generování souboru v CSV, neboť nebyla potřeba implementovat knihovnu a používané metody byly snadno pochopitelné. Při generování XML bylo již zapotřebí vyhledat externí knihovnu a daná velikost jednotlivých skriptů rostla vhledem ke složitosti. U formátů MS Word a MS Excel byl k dispozici již zmíněný balík knihoven, ale nebyl již tak snadno uchopitelný jako v předchozích případech. Knihovna vyžadovala důkladně prostudovat její dokumentaci a občas skripty vypisovaly chybové reporty. S tímto faktem rostla

i složitost samotných skriptů. Proto tyto dva formáty nabyly nižšího hodnocení. V poslední řadě byl formát PDF, kde byly k dispozici zvlášť specializované knihovny, které zahrnovaly ještě obsáhlejší dokumentace, než knihovny předchozích formátů. PDF bylo nejobtížnější vygenerovat, neboť bylo velice problematické a ve spoustě případů kolizní. Dosáhlo tedy nejnižšího hodnocení.

2.3.3.2 Aplikace metody pro malý obsah dat

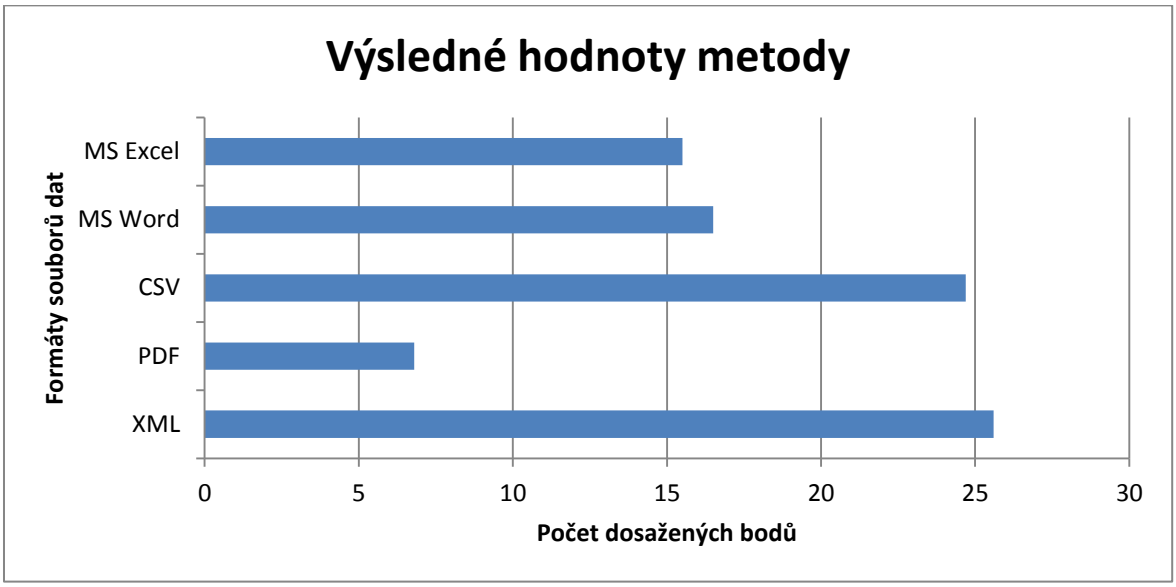
váha	vlastnosti\formáty souborů	XML	PDF	CSV	MS Word	MS Excel
1	rychlost vygenerování	$10_{(10)}$	$1_{(1)}$	$6_{(6)}$	$4_{(4)}$	$3_{(3)}$
0,9	chybovost při generování	$8_{(7,2)}$	$4_{(3,6)}$	$9_{(8,1)}$	$7_{(6,3)}$	$7_{(6,3)}$
0,7	podpora knihoven	$8_{(5,6)}$	$2_{(1,4)}$	$10_{(7)}$	$6_{(4,2)}$	$6_{(4,2)}$
0,4	složitost jazyka	$7_{(2,8)}$	$2_{(0,8)}$	$9_{(3,6)}$	$5_{(2)}$	$5_{(2)}$
		25,6	6,8	24,7	16,5	15,5

Tabulka 14: Metoda vážených kritérií pro generování souborů v jazyce PHP při malém množství dat [vlastní zpracování]



Graf 5: Metoda vážených kritérií pro generované soubory v jazyce PHP při malém množství dat [vlastní zpracování]

2.3.3.3 Výsledky metody pro malý obsah dat



Graf 6: Výsledky metody vážených kritérií pro generované soubory při malém množství dat [vlastní zpracování]

2.3.3.4 Postup při aplikování metody pro velký obsah dat

Obdobným způsobem jako u malého množství dat se postupovalo v této sekci. Zvolené 4 charakteristické vlastnosti zůstaly zachovány a změnil se pouze jejich příslušný koeficient váhy. Jelikož se jedná o generování velkých dat, tak se do popředí dostala chybovost při generování, neboť se předpokládá, že uživatel bude klást nejvyšší důraz právě na tuto vlastnost z důvodu velikého obsahu dat. Potom rychlost vygenerování nebude nadále tolik podstatná. Ostatní váhy zbylých vlastností zůstaly víceméně zachovány, změnil se důraz těchto vah z důvodu vyžadování spolehlivějších nástrojů knihoven (podpora knihoven) a optimálního rozvržení kódu jednotlivých skriptů (složitost jazyka).

Z hlediska přiřazování hodnot z hodnotící škály jednotlivým formátům se z většiny případů postupovalo shodným způsobem jako u generování malého množství dat. Jelikož byla vygenerována veliká spousta dat, tak se navyšoval počet chyb obsažených v daných souborech. Proto se vlastnost: „chybovost při generování“ opírala právě o generované soubory, které vznikly za účelem testování jako podklad této práce (jednotlivé skripty jsou k nalezení v přílohách této práce). Nejvyšší počet bodů byl zaznamenán u formátu XML, neboť se jevil zcela bezchybný. Akorát někdy vykazoval špatnou schopnost uspořádání jednotlivých prvků stromové struktury a výsledný soubor se pak stal málo přehledným. Z tohoto důvodu bylo tomuto formátu přiděleno 9 bodů. Při pozorování CSV se došlo k závěru, že tento formát měl občas obtíže

vygenerováním velkého rozsahu dat, proto se umístil až jako druhý. U formátu MS Word byla místy zaznamenána chybovost v oblasti diakritiky, ale z jiného úhlu pohledu byla vygenerovaná data zcela v pořádku. Na posledních pozicích se umístily formáty MS Excel a PDF. MS Excel při velkém množství dat měl potíže s generováním tabulek a jiných objektů. Např. tabulky byly nevhodně sestavené a obsažená data v buňkách přesahovaly její okraje. U formátu PDF byla situace velice podobná. Zde byla akorát zaznamenána vyšší chybovost při generování objektů, než u předchozího formátu.

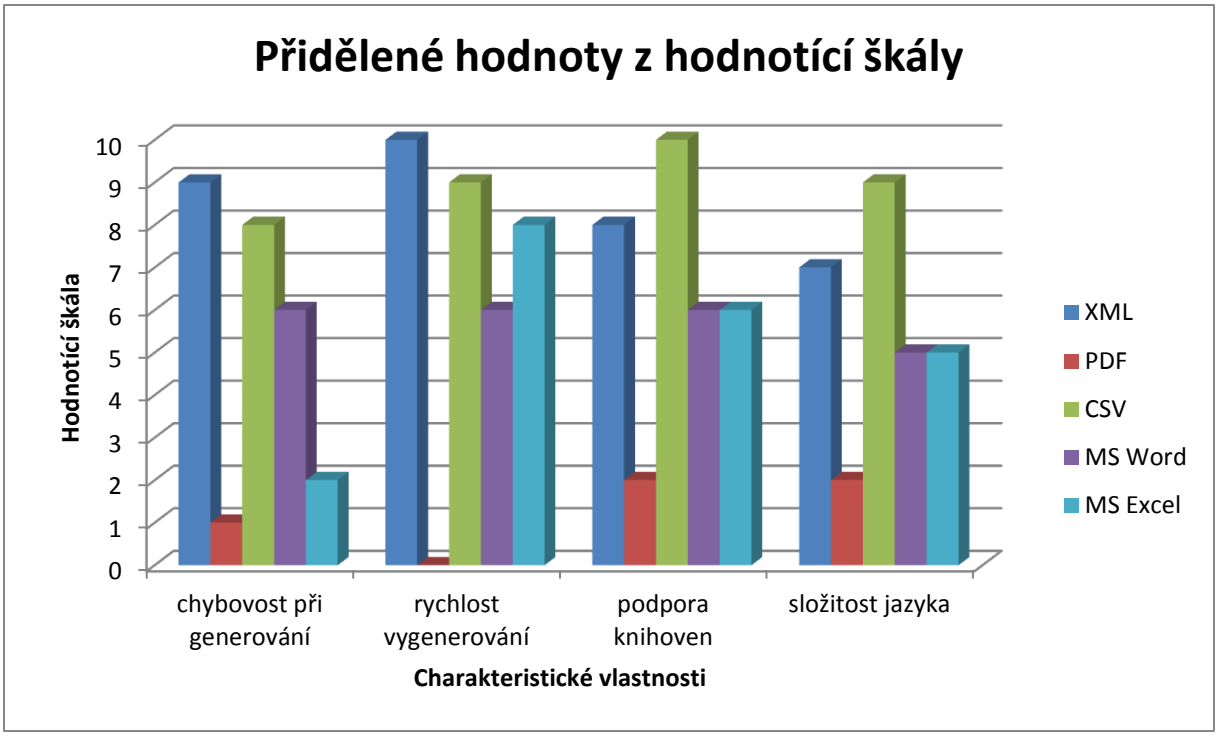
Při hodnocení rychlosti vygenerování bylo využito údajů z měření rychlosti generování souborů v jazyce PHP při velkém množství dat, které jsou uvedeny výše. Byl vytvořen poměr naměřených dat v milisekundách a ten byl následně vepsán do tabulky. Zajímavě vysoká hodnota byla naměřena u generování souboru PDF, při zahájení generování byl localhostový server přetížen a trvalo značnou dobu, než se soubor vygeneroval.

Vlastnosti: „podpora knihoven“ a „složitost jazyka“ byly již analyzovány při testování generování malého obsahu dat, proto zde byly zaznamenány shodné hodnoty.

2.3.3.5 Aplikace metody pro velký obsah dat

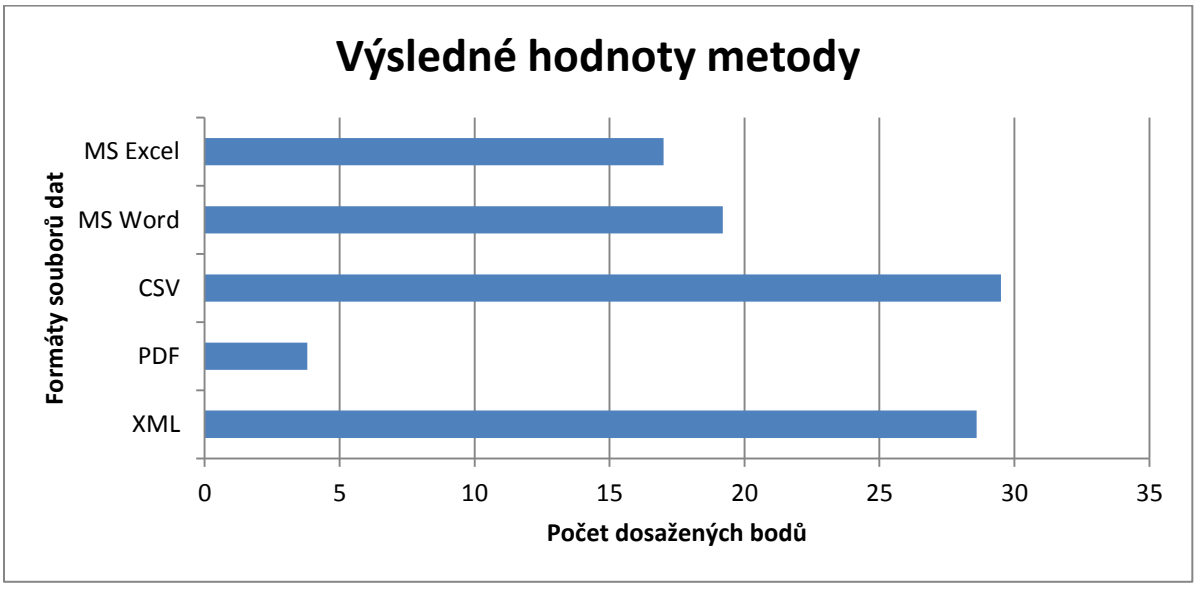
váha	vlastnosti\formáty souborů	XML	PDF	CSV	MS Word	MS Excel
1	chybovost při generování	$9_{(9)}$	$1_{(1)}$	$8_{(8)}$	$6_{(6)}$	$2_{(2)}$
0,9	rychlost vygenerování	$10_{(9)}$	$0_{(0)}$	$9_{(8,1)}$	$6_{(5,4)}$	$8_{(7,2)}$
0,8	podpora knihoven	$8_{(6,4)}$	$2_{(1,6)}$	$10_{(8)}$	$6_{(4,8)}$	$6_{(4,8)}$
0,6	složitost jazyka	$7_{(4,2)}$	$2_{(1,2)}$	$9_{(5,4)}$	$5_{(3)}$	$5_{(3)}$
		28,6	3,8	29,5	19,2	17

Tabulka 15: Metoda vážených kritérií pro generování souborů v jazyce PHP při velkém množství dat [vlastní zpracování]



Graf 7: Metoda vážených kritérií pro generované soubory v jazyce PHP při velkém množství dat [vlastní zpracování]

2.3.3.6 Výsledky metody pro velký obsah dat



Graf 8: Výsledky metody vážených kritérií pro generované soubory v jazyce PHP při velkém množství dat [vlastní zpracování]

2.3.4 Metoda vážených kritérií pro export dat v jazyce Java do různých formátů souborů

2.3.4.1 Postup při aplikování metody pro malý obsah dat

Z hlediska testovaných vlastností bylo použito stejných vah jako při generování souborů v jazyce PHP při malém množství dat, které jsou uvedeny v příslušné sekci daného jazyka. Bylo tak učiněno, protože je dbáno především na požadavky dané aplikace a ne na jazyk, pomocí kterého je zkompletována.

Nyní k samotnému hodnocení formátů souborů dat. Pro vlastnost: „rychlost vygenerování“ bylo opět užito výsledných hodnot z měření rychlosti generování souborů v jazyce Java při malém množství dat, které je uvedeno výše. Naměřené údaje v milisekundách byly poměrově porovnány a následně zaznamenány do tabulky.

Při testování chybovosti při generování se analyzovaly jednotlivé exportované soubory. Jelikož se jednalo o generování malých dat, tak se především vycházelo z poznatků teoretické části této práce. Všechny formáty se umístily v hodnocení relativně přijatelným způsobem. Formáty XML a CSV byly naprosto bezchybné, proto nabyly nejvyššího hodnocení. U formátu MS Excel byla zaznamenána chyba při generování data a času, kde daný formát nemohl být přečten za pomoci specializovaného programu (Microsoft Excel). Zajímavým poznatkem bylo generování souboru PDF. Data zde byla velice vhodným způsobem uspořádána a celkový soubor tvořil velice přijatelnou grafickou podobu. Dalo by se říci, že jazyk si při generování souboru PDF při malém množství dat vedl lépe, než předchozí testovaný jazyk v této oblasti. Avšak cílem této práce nebylo vzájemné porovnání těchto dvou jazyků.

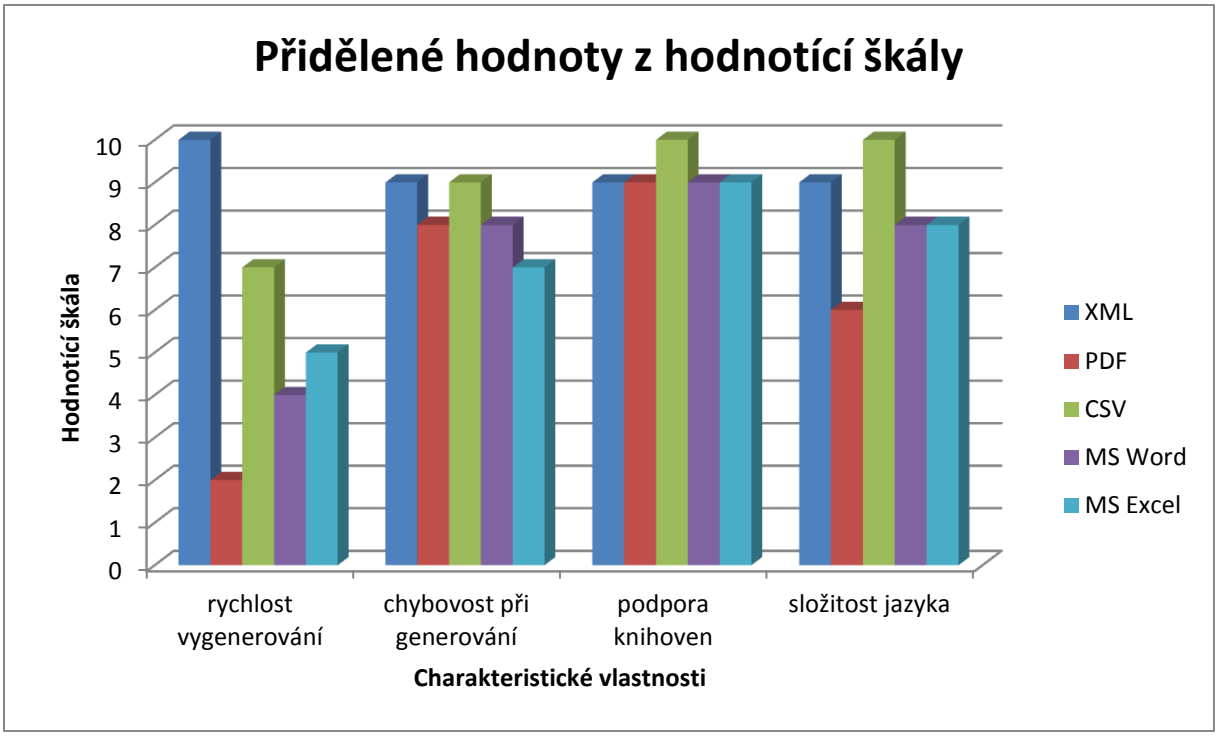
Při hodnocení podpory knihoven se analyzovaly všechny možné dostupné knihovny, které sloužily právě pro generování příslušných souborů. Dále se posuzovala složitost použití jednotlivých vybraných knihoven. Nejlepší hodnocení obdržel formát CSV, neboť příslušné knihovny byly již implementovány ve vývojovém prostředí (NetBeans IDE 8.0.1) a práce s nimi byla velice jednoduchá. Dále u jazyka Java byl export dat do různých formátů souborů velice podpořen z hlediska knihoven. Knihovny byly snadno uchopitelné a nevykazovaly vysokou chybovost. Např. pro většinu exportování zbylých formátů tak mohla být jednoduše použita sada knihoven zvaných: „POI“ (od Apache Software Foundation). Z těchto důvodů bylo hodnocení těchto zbylých formátů porovnatelné.

Z hlediska složitosti jazyka se analyzovala celková přehlednost kódu jednotlivých tříd, odhadnutelnost používaných metod a způsob, kterým jazyk Java dokáže uchopit generování příslušného souboru s daným formátem dat. Nejlépe se umístil formát CSV, kde jednotlivé třídy byly obsahově minimální a zdáli se jako nejsnadněji pochopitelné. Složitější pak již byl XML, u kterého se pracovalo s instancemi továren pro sestavování souborů. Z hlediska kódu u formátů MS Word a MS Excel nebylo použití metody příliš komplikované, zvlášť když uživatel potřeboval vygenerovat pouze malé množství dat. Nicméně se zde pracovalo se specializovanou knihovnou a bylo zapotřebí studie příslušné dokumentace. Proto tyto dva formáty nabyly o jednotku nižší hodnocení. Jako poslední v hodnocení byl formát PDF, neboť složitost metod zde narůstala a v mnoha případech se nedařilo daná data vygenerovat, jelikož kód pro generování tohoto formátu měl pevně danou strukturu, kterou bylo nutné dodržet. Z tohoto důvodu byl formát PDF ohodnocen nejhůře.

2.3.4.2 Aplikace metody pro malý obsah dat

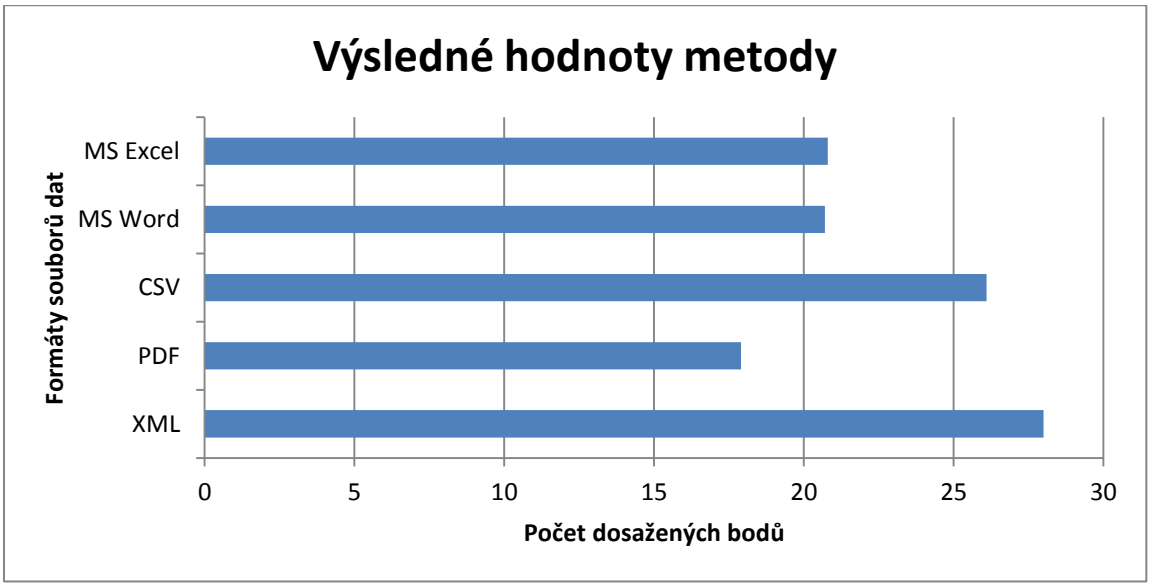
váha	vlastnosti\formáty souborů	XML	PDF	CSV	MS Word	MS Excel
1	rychlost vygenerování	$10_{(10)}$	$2_{(2)}$	$7_{(7)}$	$4_{(4)}$	$5_{(5)}$
0,9	chybovost při generování	$9_{(8,1)}$	$8_{(7,2)}$	$9_{(8,1)}$	$8_{(7,2)}$	$7_{(6,3)}$
0,7	podpora knihoven	$9_{(6,3)}$	$9_{(6,3)}$	$10_{(7)}$	$9_{(6,3)}$	$9_{(6,3)}$
0,4	složitost jazyka	$9_{(3,6)}$	$6_{(2,4)}$	$10_{(4)}$	$8_{(3,2)}$	$8_{(3,2)}$
		28	17,9	26,1	20,7	20,8

Tabulka 16: Metoda vážených kritérií pro generování souborů v jazyce Java při malém množství dat [vlastní zpracování]



Graf 9: Metoda vážených kritérií pro generované soubory v jazyce Java při malém množství dat [vlastní zpracování]

2.3.4.3 Výsledky metody pro malý obsah dat



Graf 10: Výsledky metody vážených kritérií pro generované soubory v jazyce Java při malém množství dat [vlastní zpracování]

2.3.4.4 Postup při aplikování metody pro velký obsah dat

Při určení vah jednotlivých vlastností se postupovalo stejným způsobem jako při generování souborů v jazyce PHP při velkém množství dat. Důvody tohoto postupu již byly zmíněny u metody pro malé množství dat.

Při hodnocení testovaných formátů se postupovalo obdobným způsobem jako u malých dat tohoto jazyka. Z hlediska vlastnosti: „chybovost při generování" se dbalo na jednotlivé vygenerované soubory. Jelikož se jednalo o velké množství dat, tak bylo pravidlem, že s rostoucím obsahem roste i potencionální výskyt chyb. Dále se hodnocení této vlastnosti opíralo o poznatky z teoretické části této práce. Formát XML byl vyhodnocen jako přívětivější, neboť oproti CSV tento formát byl ucelený (bylo jednoznačně rozpoznatelné, kde končí a kde začíná). Výrazně v hodnocení poklesl formát MS Excel (oproti malému množství dat) a to právě z důvodu velkých dat. Při generování rozsáhlejšího dokumentu zde chybovost rostla exponenciálně. Jednalo se především o chybný formát data a času, nevhodně generované tabulky, apod. U PDF byla situace velice podobná, při generování se vyskytlo mnoho chyb a ještě samotný proces byl vysoce náročný (oproti ostatním formátům) pro výpočetní jednotku. Proto generování do těchto dvou formátů souborů nabylo nízkého hodnocení.

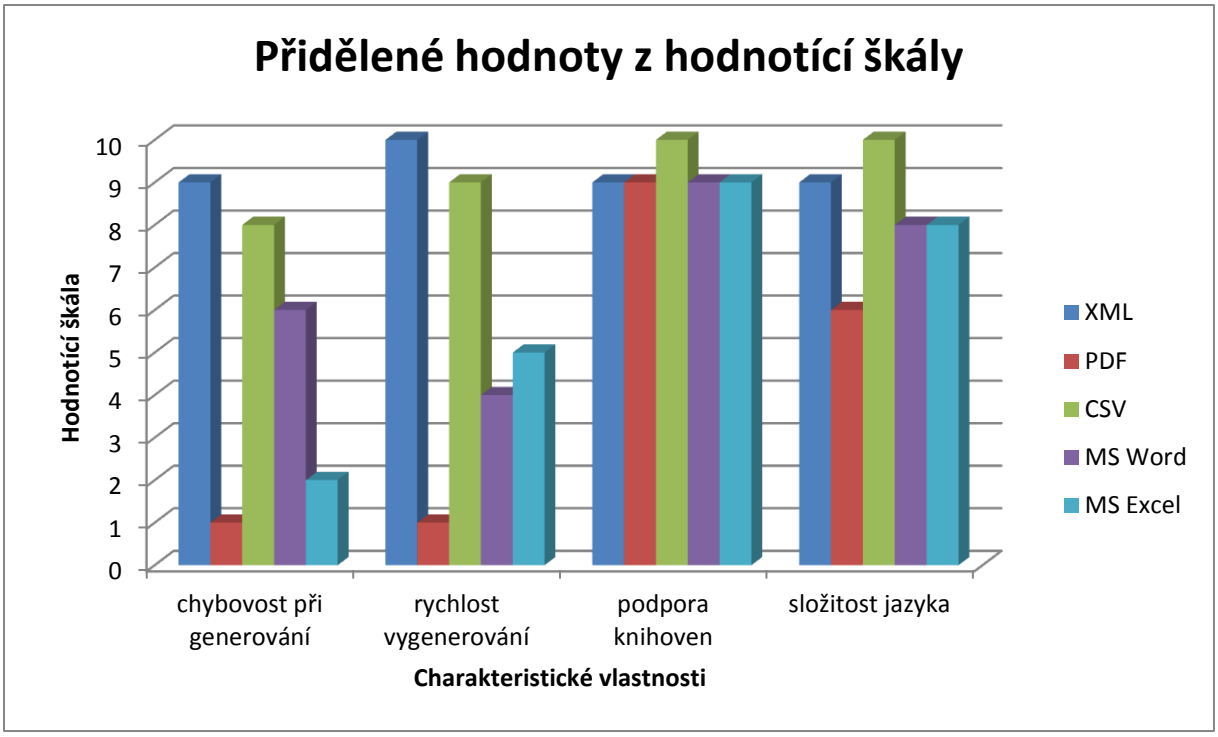
Další vlastnost: „rychlost vygenerování" byla souzena opět podle výsledků z měření rychlosti generování souborů v jazyce Java při velkém množství dat. Metodika měření je uvedena v jedné z předchozích kapitol.

Zbylé dvě vlastnosti: „podpora knihoven" a „složitost jazyka" byly hodnoceny stejným způsobem jako při testování malé množiny dat. Postup při analýze těchto hodnot je k nalezení právě v kapitole pojednávající o generování souborů v jazyce Java při malém množství dat.

2.3.4.5 Aplikace metody pro velký obsah dat

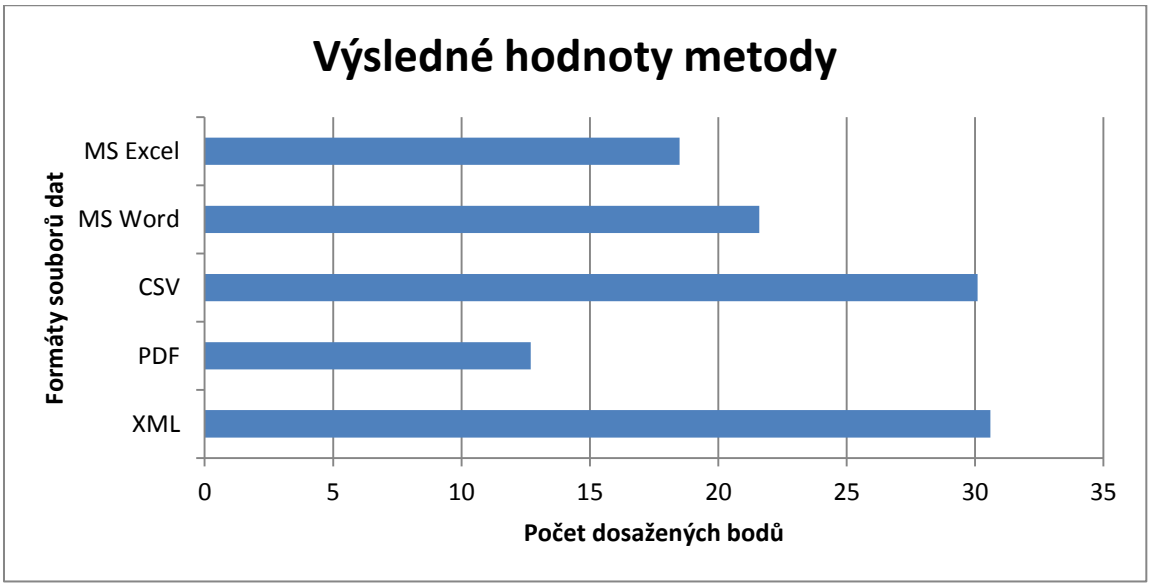
váha	vlastnosti\formáty souborů	XML	PDF	CSV	MS Word	MS Excel
1	chybovost při generování	$9_{(9)}$	$1_{(1)}$	$8_{(8)}$	$6_{(6)}$	$2_{(2)}$
0,9	rychlost vygenerování	$10_{(9)}$	$1_{(0,9)}$	$9_{(8,1)}$	$4_{(3,6)}$	$5_{(4,5)}$
0,8	podpora knihoven	$9_{(7,2)}$	$9_{(7,2)}$	$10_{(8)}$	$9_{(7,2)}$	$9_{(7,2)}$
0,6	složitost jazyka	$9_{(5,4)}$	$6_{(3,6)}$	$10_{(6)}$	$8_{(4,8)}$	$8_{(4,8)}$
		30,6	12,7	30,1	21,6	18,5

Tabulka 17: Metoda vážených kritérií pro generování souborů v jazyce Java při velkém množství dat [vlastní zpracování]



Graf 11: Metoda vážených kritérií pro generované soubory v jazyce Java při velkém množství dat [vlastní zpracování]

2.3.4.6 Výsledky metody pro velký obsah dat



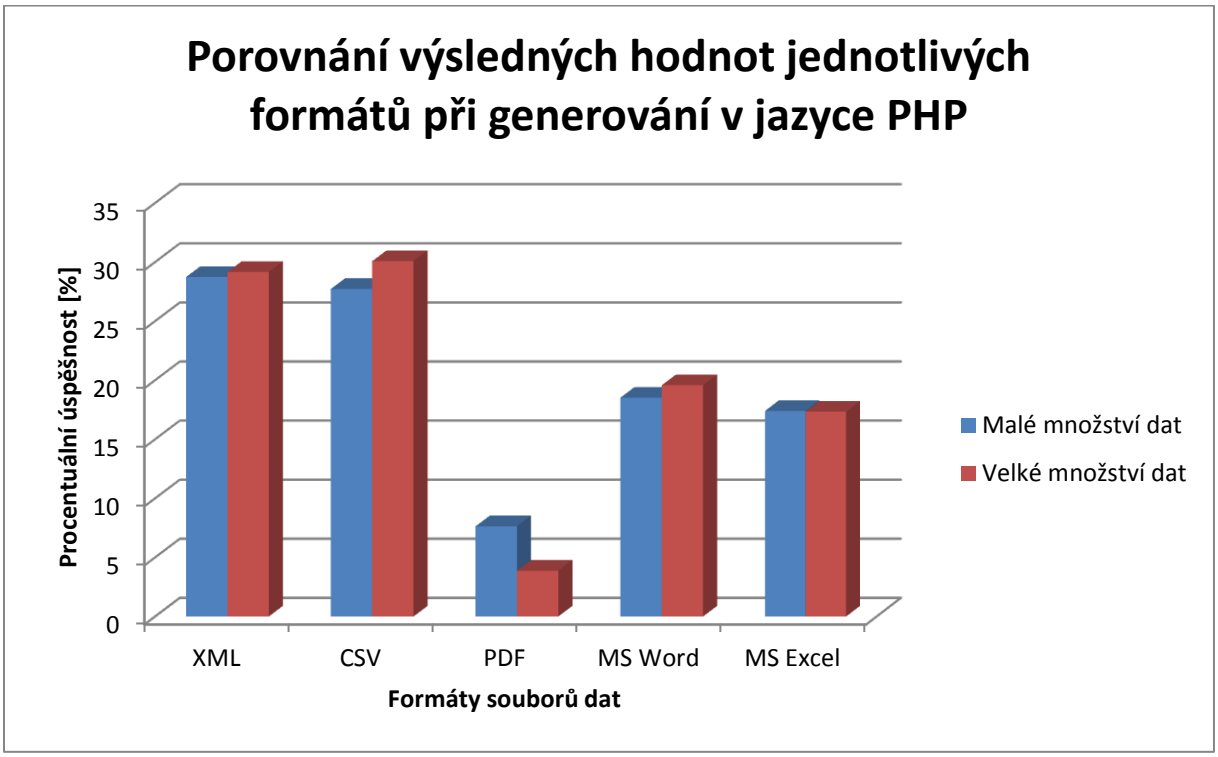
Graf 12: Výsledky metody vážených kritérií pro generované soubory v jazyce Java při velkém množství dat [vlastní zpracování]

3 Shrnutí výsledků práce

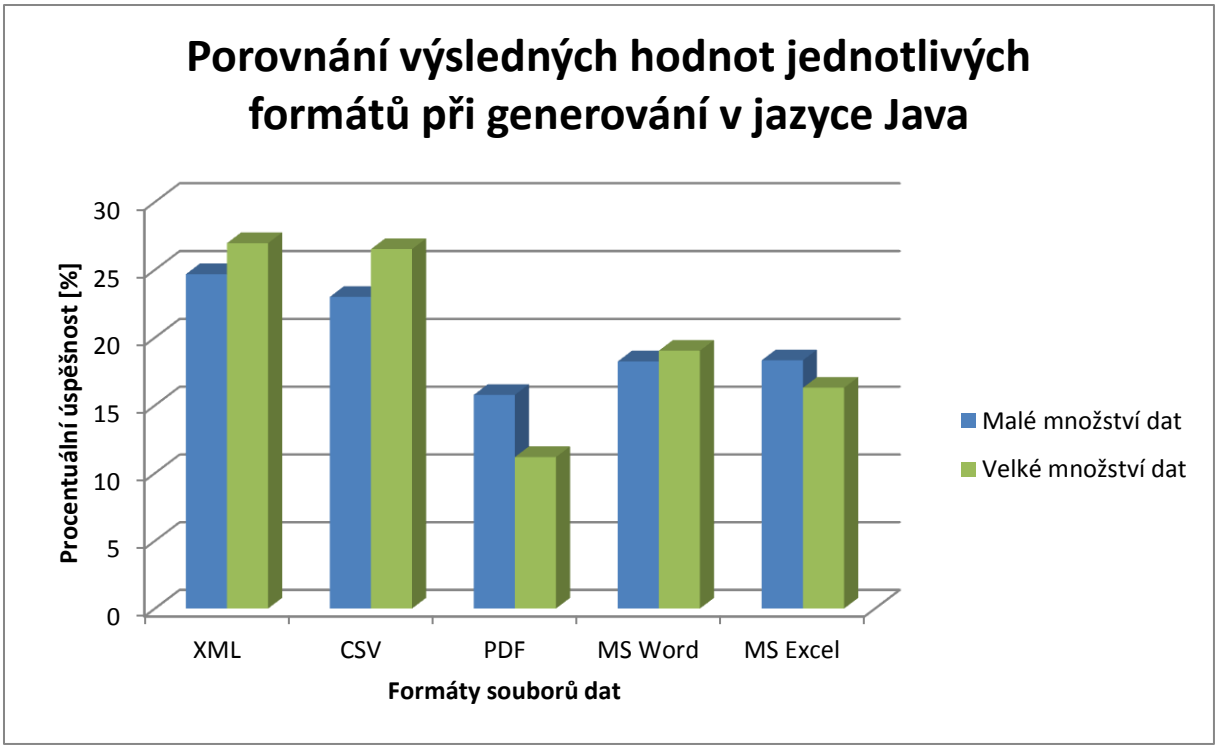
Následující grafy byly zkonstruovány na základě výsledných hodnot jednotlivých metod vážených kritérií. Byl proveden celkový součet všech výsledných hodnot pro danou sekci a z této sumy se následně vypočítal procentuální podíl z celkového součtu. Tento postup byl proveden pro malé i velké množství dat. Použilo se tohoto typu grafického vyjádření, neboť bylo vhodné pro porovnání jednotlivých formátů mezi sebou. Metodika tvorby těchto grafů je zahrnuta v souboru, který se nachází v přílohách této práce.

Popis těchto výsledků a vhodné doporučení v oblasti navržené aplikace jsou uvedeny v samotném závěru této práce.

3.1 Generování souborů pomocí jazyků

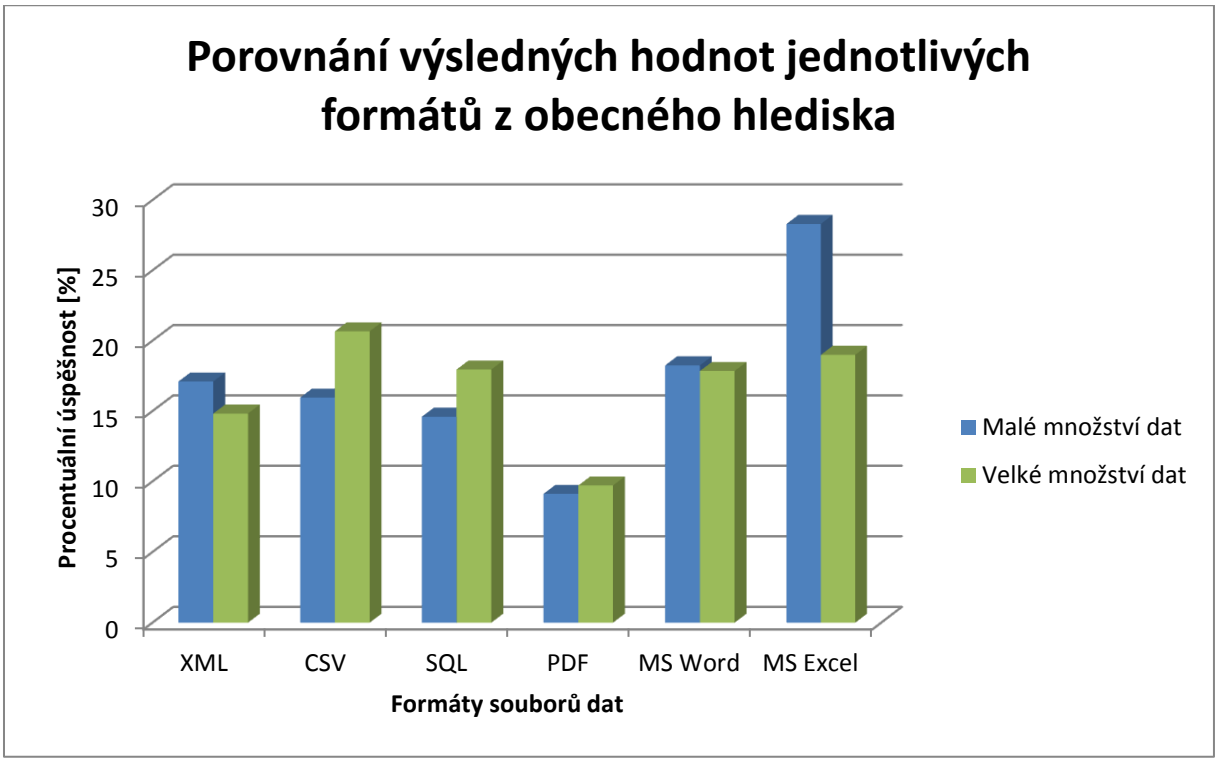


Graf 13: Procentuální vyjádření úspěšnosti jednotlivých formátů pro generování pomocí jazyka PHP [vlastní zpracování]



Graf 14: Procentuální vyjádření úspěšnosti jednotlivých formátů pro generování pomocí jazyka Java [vlastní zpracování]

3.2 Obecné charakteristiky formátů souborů



Graf 15: Procentuální vyjádření úspěšnosti jednotlivých formátů pro jejich obecné charakteristiky [vlastní zpracování]

Závěr

Celá tato práce se zabývá důležitými rysy a charakteristikami různých výstupních formátů souborů dat. V počáteční fázi byly analyzovány jednotlivé formáty souborů z co nejširšího úhlu pohledu, dbal se důraz především na jejich výhody a nevýhody, kde byla rozebírána struktura formátů, jakým způsobem se formáty používají, jejich vzhled, kompatibilita verzí, apod. Tato analýza nebyla podložena pouze za pomoci odborných textů, ale přispěla i názorná ukázka miniaturní databáze, která byla exportována do všech zmíněných výstupních formátů.

Následovala praktická část, jejíž hlavním cílem bylo navrhnutí optimální kombinace výstupního formátu souboru a množství dat, které by mělo být exportováno, pro základy aplikací, které se zabývají generováním různých souborů s příslušnými formáty dat. Z hlediska používání by tato aplikace přijala data od uživatele, analyzovala by jejich délku a na základě analýzy by uživateli doporučila, který výstupní formát by byl pro tato data optimální.

Bylo provedeno několik výpočtů pomocí metody vážených kritérií, která byla zaměřena na logickou (rozhodovací) složku již zmíněné aplikace. Byly vybrány 2 programovací jazyky, pomocí nichž by mohla být aplikace sestavena. Jednalo se o vybrané jazyky PHP a Javu z toho důvodu, že se jedná o veliký trend v této době a je po nich značná poptávka v oblasti tvorby aplikací. Cílem samotných výpočtů by mělo být navrhnutí optimálního výstupního formátu dat při velkém a malém množství při základě daného programovacího jazyka.

Z obecného hlediska při zkoumání jednotlivých výstupních formátů dat se došlo k tomuto závěru. Při exportování nebo při samotném užití z hlediska práce s daty a jejich formáty se u malého množství dat se doporučuje využít souborového formátu CSV určený pro MS Excel. Je to z toho důvodu, neboť se jedná o velice jednoduchý formát z hlediska struktury a tento fakt může být efektivní, pokud se jedná o rychlost zpracování za pomoci nějakého softwaru (např. textového editoru). V tomto případě jsou data strukturována a exportována pro daný tabulkový procesor (Microsoft Excel).
Tento formát si vedl velice přívětivým způsobem u rychlosti exportování daného souboru (169 ms), u velikosti souboru, kde byla zaznamenána velice nízká hodnota (1 kB), z hlediska složitosti a dalších vlastností byl také přívětivý (doloženo v praktické části). Jako málo vhodným formátem v tomto kritériu se stalo PDF. Tento formát byl nepříznivý z hlediska rychlosti exportu (prioritní vlastnost pro malý obsah dat), kde čas

jeho exportování nabýval nejvyšší hodnoty (397 ms) oproti ostatním formátům. Jelikož bylo potřeba vyexportovat mnoho prvků, které tento formát zahrnuje, tak se rapidním způsobem navýšila jeho velikost (179 kB). U ostatních charakteristik se vedl lépe (viz praktická část). Ostatní formáty souborů dat byly vyhodnoceny podobnou mírou. Každých z těchto formátů nesl jak pozitivní, tak negativní hodnoty při hodnocení. Důvodem se stala jejich různorodost a nejvíce rozhodovací složkou této sekce se stala právě rychlost exportu (viz praktická část).

V další části došlo ke zkoumání nejvhodnějšího výstupního formátu dat při exportování velkého množství dat. V průzkumu se ukázalo, že optimální pro velká data by byl jazyk CSV. V této sekci se prioritním kritériem stala velikost vyexportovaného souboru, kde tento formát zaznamenal nejnižší hodnotu (29 kB) oproti ostatním formátům (stejná hodnota byla naměřena i u CSV určeného pro MS Excel, ale v potaz se bral základ tohoto formátu). Opět (jako u malého množství dat) byla velice pozitivním způsobem hodnocena náročnost tohoto formátu (viz praktická část) a z hlediska rychlosti exportování si vedl také dobře (312 ms). Pokud bude přihlédnuto na nejméně vhodný formát pro exportování velkého množství dat, tak nízké výsledné hodnoty byly zaznamenány u formátu PDF. Při rychlosti exportování tohoto souboru byla naměřena kriticky vysoká hodnota (2270 ms). Oproti malému množství dat tento formát neskončil na posledním místě u zkoumání velikosti souboru, ale také nabýval vyšší hodnoty (170 kB), která měla negativní vliv na výsledné hodnoty. U ostatních vlastností si vedl lépe (viz praktická část). Další formáty souborů dat opět nabývaly podobných hodnot, nebyl zde zaznamenán tak vysoký rozdíl oproti optimálnímu vyhodnocenému formátu (CSV), proto jejich použití pro export velkého množství dat není možné zcela nedoporučit. Informace o silných a slabých stranách všech formátů jsou k nalezení v praktické části této práce.

V úvahu, že zkoumaná aplikace bude naprogramována v jazyce PHP, se došlo k následujícímu závěru. Pro malé množství dat se nejvhodnějším generovaným formátem stal XML. Tento formát byl vygenerován za pouhých 12 milisekund, a jelikož rychlost generování souboru je u malého obsahu dat velice podstatná, tak se stal optimálním. CSV se umístilo hned na druhém místě a to bylo zapříčiněno především díky již zmíněné rychlosti generování, kde tento soubor byl vyhotoven za 340 milisekund. Pro malé množství dat by podle výsledků nebylo doporučeno využít formát PDF, neboť byl vygenerován za 917 milisekund, což je podstatně více oproti

ostatním zkoumaným formátům, obsahoval mnohdy chyby (překrývání textu, překrývání různorodých elementů apod.) a knihovna, která podporovala tvorbu tohoto formátu, byla vyhodnocena jako nejsložitější z hlediska rozsahu a samotného použití.

Z pohledu velkého množství dat by vygenerování pomocí aplikace sestavené na základě jazyka PHP bylo optimální pro CSV. Tento formát byl zmíněn již v mnoha souvislostech a jazyk PHP s ním dokáže velice efektivně pracovat. Samotný soubor CSV s velkým množstvím dat byl vygenerován za 350 milisekund, ale u rozsáhlejších dat se nedbalo tolik na rychlost vygenerování samotného souboru jako na výskyt chyb při generování, proto se formát XML dostal až na druhou pozici. Značkovací jazyk XML neměl tolik rozmanitou možnost použití knihoven, ale použití bylo velice snadné. Rychlost vygenerování souboru s formátem XML byla znatelně nejnižší: 21 milisekund. Jako nejméně vhodný formát za této situace se stal PDF. Všechny jeho měřené vlastnosti dosahovaly kriticky malých hodnot, dokonce při generování byl často přetížen localhostový server. Hrubý odhad rychlosti vygenerování souboru v PDF při velkém obsahu dat byl 18660 milisekund.

Za předpokladu, že aplikace pro exportování dat bude naprogramována v jazyce Java, byla zjištěna tato fakta. Opět bylo zkoumáno velké a malé množství dat exportování. Při malém obsahu dat bylo zjištěno, že vhodným výstupním formátem dat by se stal XML. Tento již několikrát zmíněný formát se dostal do popředí především díky rychlost vygenerování: 19 milisekund. U ostatních vlastností si vedl také velice dobře, a proto se mu dalo jen stěží konkurovat. CSV sice mělo velice dobré výsledky u podpory knihoven (neboť používaný software má již v základu implementovánu knihovnu pro exportování dat do CSV) a u složitosti (z hlediska struktury je nejjednodušší), ale vygenerování samotného souboru trvalo 112 milisekund. Málo optimální se v této oblasti stal formát PDF. Podpora knihoven zde byla na lepší úrovni, než u jazyka PHP, ale opět vygenerování takového souboru bylo příliš dlouhé. Čas vygenerování souboru PDF s malým množstvím dat byl zhruba 495 milisekund.

Následovalo vyhodnocení pro generování souborů v jazyce Java pro velké množství dat. Na předních příčkách vhodných formátů vyexportovaných souborů pro tuto problematiku se umístily hned 2 formáty. XML překonal CSV o pouhou polovinu procenta, neboť měl menší výskyt chyb (např. nadbytečné mezery) a to bylo hlavní prioritou při exportování velkého obnosu dat. CSV měl zase silnou podporu z hlediska

implementovaných knihoven, již zmíněných v předchozím odstavci, a také získal hodně bodů díky jeho jednoduchosti. Jako nevhodný formát se znovu stal PDF. Opět se soubor tohoto formátu generoval velice dlouho (5600 milisekund) a obsahoval veliké množství chyb.

Úplným závěrem této práce by se mohlo polemizovat o zkoumaných výstupních formátech souborů dat této práce. Pokud by se jednalo o práci s exportováním malého množství dat, tak by se na základě výsledků mohlo soudit, že souborový formát CSV je velice jednoduchým a lehce uchopitelným nástrojem pro práci s daty. Ve většině případů nevyžaduje připojení externích knihoven, neboť obvykle tyto knihovny bývají již implementovány ve vývojových prostředích (např. NetBeans, Eclipse). Dále uživatel nepotřebuje mnoho zkušeností, aby mohl tento nástroj používat. Jelikož má jednoduchou strukturu, tak nebývá problém otevřít tento formát ve veliké spoustě programů.

Při práci s velkým množstvím dat se dá spekulovat, že by bylo vhodné jak použití formátu CSV, tak XML. Každý z těchto formátů nabízí různorodé charakteristiky, které by se daly využít velice širokou škálou způsobů. Proto by záleželo na samotném uživateli a na jeho účelech s daty, se kterými by chtěl manipulovat.

Seznam použité literatury

Tištěné zdroje

[1] CASTRO, Elizabeth. XML pro World Wide Web. Praha: SoftPress, c2001, 254 s. Praktická vizuální příručka. ISBN 80-86497-07-0.

[2] CONNOLLY, Thomas M a Carolyn E BEGG. Database systems: a practical approach to design, implementation, and management. 4th ed. New York: Addison-Wesley, c2005, l, 1374 p. ISBN 0321210255.

[3] CONOLLY, Thomas, Carolyn E BEGG a Richard HOLOWCZAK. Mistrovství - databáze: profesionální průvodce tvorbou efektivních databází. Vyd. 1. Brno: Computer Press, 2009, 584 s. ISBN 978-80-251-2328-7.

[4] HAROLD, Elliotte Rusty a W MEANS. XML v kostce: pohotová referenční příručka v kostce. Vyd. 1. Praha: Computer Press, 2002, xvi, 439 s. ISBN 80-7226-712-4.

[5] KOSEK, Jiří. XML pro každého: podrobný průvodce. 1. vyd. Praha: Grada, 2000, 163 s. ISBN 80-7169-860-1.

Ostatní zdroje

[6] ADOBE SYSTEMS. Soubory PDF, formát Adobe PDF | Adobe Acrobat XI [online]. 2015 [cit. 2015-01-26]. Dostupné z: http://www.adobe.com/cz/products/acrobat/adobepdf.html

[7] ANDERSON, Ty a HART-DAVIS. Beginning Microsoft Word 2010 [online]. 2010 [cit. 2015-04-18]. ISBN 978-1-4302-2953-7. Dostupné z: https://books.google.cz/books?id=HNX2snz76UIC&pg=PA14&lpg=PA14&dq=ty+anderson+hart-davis+Beginning+Microsoft+Word+2010+pdf&source=bl&ots=v3b_tdVEeF&sig=Vmrd9UVVt8ydIHS5KTN_V15hsFE&hl=cs&sa=X&ei=d8EyVZXcLMjcaO6ugOAK&ved=0CDAQ6AEwAg#v=onepage&q=ty%20anderson%20hart-davis%20Beginning%20Microsoft%20Word%202010%20pdf&f=false

[8] ASPIN, Adam. SQL Server 2012 data integration recipes [online]. S.l.: Apress, 2012 [cit. 2015-03-26]. ISBN 978-1-4302-4791-3. Dostupné z: http://www.mvatcybernet.com/IT%20E-BOOKS/IT%20PDF%20Books/IT%20BOOKS/MICROSOFT%20BOOKS/SQL%202012/SQL%20SERVER%202012%20DATA%20INTEGRATION%20RECIPES.pdf

[9] DOYLE, Matt. Beginning PHP 5.3 [online]. Indianapolis, IN: Wiley Pub., c2010, xxxiv, 801 p. [cit. 2015-04-18]. Wrox beginning guides. ISBN 04-704-1396-4. Dostupné z: http://www.cs.karelia.ru/~musen/php/2013/Books/Beginning%20PHP%205.3%20by%20Matt%20Doyle.pdf

[10] FRIEDRICH, Matthias. A Metadata Format For CSV Files [online]. 2015 [cit. 2015-05-02]. Dostupné z: http://blog.mafr.de/2008/04/26/metadata-for-csv/

[11] FRYE, Curtis. Microsoft Excel 2010: step by step [online]. Redmond, Wash.: Microsoft Press, c2010, xli, 436 p. [cit. 2015-04-18]. ISBN 07-356-2694-4. Dostupné z: http://www.seu.ac.lk/fac/student%20download/MS%20Excel%202010%20Step%20by%20step.pdf

[12] HW SERVER S.R.O. PDF - přenosný formát dokumentů | HW.cz [online]. 2014 [cit. 2015-01-26]. Dostupné z: http://www.hw.cz/teorie-a-praxe/dokumentace/pdf-prenosny-format-dokumentu.html

[13] ITeach [online]. 2015 [cit. 2015-04-12]. Dostupné z: http://project67555.appspot.com/sql.html

[14] KEPRT, Aleš. XML [online]. 2005 [cit. 2015-01-28]. Dostupné z: http://www.keprt.cz/texty/xml.pdf

[15] MySQL :: MySQL 5.0 Reference Manual :: 13.1.6 CREATE DATABASE Syntax [online]. 2015 [cit. 2015-01-26]. Dostupné z: http://dev.mysql.com/doc/refman/5.0/en/create-database.html

[16] NIXON, Robin. Learning PHP, MySQL & JavaScript: WITH JQUERY, CSS & HTML5 [online]. 2012 [cit. 2015-04-18]. ISBN 978-1-491-91866-1. Dostupné z: http://cdn.oreillystatic.com/oreilly/booksamplers/9781491918661_sampler.pdf

[17] QUINSTREET ENTERPRISE. SQLCourse - Interactive Online SQL Training for Beginners [online]. 2015 [cit. 2015-01-26]. Dostupné z: http://www.sqlcourse.com/

[18] SHARAN, Kishory. Scripting in Java: Integrating with Groovy and Javascript [online]. 2014 [cit. 2015-04-12]. ISBN 978-1-484207-14-7. Dostupné z: http://it-ebooks.info/book/4525/

[19] SQL INSERT INTO Statement [online]. 2015 [cit. 2015-01-26]. Dostupné z: http://www.w3schools.com/sql/sql_insert.asp

[20] SQL: kompletní kapesní průvodce [online]. 1999 [cit. 2015-01-26]. ISBN 80-7169-692-7. Dostupné z: http://nb.vse.cz/~simunek/files/KKP_SQL.pdf

[21] URMA, Raoul-Gabriel, Mario FUSCO a Alan MYCROFT. Java 8 in action: lambdas, streams, and functional-style programming [online]. Shelter Island, NY: Manning, c2015, xxviii, 394 s. [cit. 2015-04-18]. ISBN 978-1-617291-99-9. Dostupné z: https://s3.amazonaws.com/mylekhaebook/book/IT+%26+Programming/Java/Java+8+in+Action.pdf

[22] W3SCHOOLS.COM. DTD Tutorial [online]. 2015 [cit. 2015-05-02]. Dostupné z: http://www.w3schools.com/dtd/

[23] WEBDESIGN - ADAPTIC. Tvorba webu [online]. 2008 [cit. 2015-01-26]. Dostupné z: http://www.tvorba-webu.cz/xml/

[24] What is Portable Document Format (PDF)? - Definition from WhatIs.com [online]. 2015 [cit. 2015-01-26]. Dostupné z: http://whatis.techtarget.com/definition/Portable-Document-Format-PDF

[25] WHITINGTON, John. PDF explained [online]. 1st ed. Sebastopol, Calif.: O'Reilly Media, 2011, c2012., xi, 121 p. [cit. 2015-04-13]. ISBN 14-493-1002-8. Dostupné z: http://it-ebooks.info/book/545/

Seznam tabulek

Seznam obrázků

Seznam grafů

Přílohy

1. Ukázkový příklad exportu tabulky adresa v XML

```
<?xml version="1.0" encoding="utf-8"?>
<!--
- phpMyAdmin XML Dump
- version 3.4.10.1
- http://www.phpmyadmin.net
-
- Počítač: localhost:3306
- Vygenerováno: Sob 24. led 2015, 16:58
- Verze MySQL: 5.5.24
- Verze PHP: 5.3.13
-->

<pma_xml_export                                                 version="1.0"
xmlns:pma="http://www.phpmyadmin.net/some_doc_url/">
    <!--
    - Structure schemas
    -->
    <pma:structure_schemas>
        <pma:database     name="suchava1"     collation="utf8_czech_ci"
charset="utf8">
            <pma:table name="Adresa">
                CREATE TABLE `Adresa` (
                  `adresa_id` int(11) NOT NULL AUTO_INCREMENT,
                  `ulice`  varchar(30)  COLLATE  utf8_czech_ci  DEFAULT
NULL,
                  `cp` varchar(10) COLLATE utf8_czech_ci DEFAULT NULL,
                  `mesto`  varchar(30)  COLLATE  utf8_czech_ci  DEFAULT
NULL,
                  `psc` char(5) COLLATE utf8_czech_ci DEFAULT NULL,
                  PRIMARY KEY (`adresa_id`)
                ) ENGINE=InnoDB AUTO_INCREMENT=12 DEFAULT CHARSET=utf8
COLLATE=utf8_czech_ci;
            </pma:table>
        </pma:database>
    </pma:structure_schemas>

    <!--
    - Databáze: 'suchava1'
    -->
    <database name="suchava1">
        <!-- Tabulka Adresa -->
        <table name="Adresa">
            <column name="adresa_id">7</column>
            <column name="ulice">Kroměřížská</column>
            <column name="cp">23</column>
            <column name="mesto">Jičín</column>
            <column name="psc">50401</column>
        </table>
        <table name="Adresa">
            <column name="adresa_id">8</column>
            <column name="ulice">U Jelena</column>
            <column name="cp">26</column>
            <column name="mesto">Pardubice</column>
            <column name="psc">50401</column>
        </table>
        <table name="Adresa">
            <column name="adresa_id">10</column>
            <column name="ulice">Družstevní</column>
```

```
            <column name="cp">113</column>
            <column name="mesto">Nový Bydžov</column>
            <column name="psc">50401</column>
        </table>
    </database>
</pma_xml_export>
```

2. Ukázkový příklad exportu tabulky adresa v SQL

```
-- phpMyAdmin SQL Dump
-- version 3.4.10.1
-- http://www.phpmyadmin.net
--
-- Počítač: localhost:3306
-- Vygenerováno: Ned 25. led 2015, 14:46
-- Verze MySQL: 5.5.24
-- Verze PHP: 5.3.13

SET SQL_MODE="NO_AUTO_VALUE_ON_ZERO";
SET time_zone = "+00:00";


/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8 */;

--
-- Databáze: `suchava1`
--

-- --------------------------------------------------------

--
-- Struktura tabulky `Adresa`
--

CREATE TABLE IF NOT EXISTS `Adresa` (
  `adresa_id` int(11) NOT NULL AUTO_INCREMENT,
  `ulice` varchar(30) COLLATE utf8_czech_ci DEFAULT NULL,
  `cp` varchar(10) COLLATE utf8_czech_ci DEFAULT NULL,
  `mesto` varchar(30) COLLATE utf8_czech_ci DEFAULT NULL,
  `psc` char(5) COLLATE utf8_czech_ci DEFAULT NULL,
  PRIMARY KEY (`adresa_id`)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8 COLLATE=utf8_czech_ci
AUTO_INCREMENT=12 ;

--
-- Vypisuji data pro tabulku `Adresa`
--

INSERT INTO `Adresa` (`adresa_id`, `ulice`, `cp`, `mesto`, `psc`) VALUES
(7, 'Kroměřížská', '23', 'Jičín', '50401'),
(8, 'U Jelena', '26', 'Pardubice', '50401'),
(10, 'Družstevní', '113', 'Nový Bydžov', '50401');

/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
```

3. Ukázkový příklad metadat MS Excel souboru zapsaných v jazyce XML

```
<Object>
    <Name>new simple</Name>
    <ID>0</ID>
    <ReferenceNumber></ReferenceNumber>
    <GroupIdentifier></GroupIdentifier>
    <PersistentIdentifier></PersistentIdentifier>
    <MasterCreationDate locale="CEST">
      <Date format="yyyyMMdd">20150501</Date>
      <Time format="HHmmssSSS">145653415</Time>
    </MasterCreationDate>
    <ObjectComposition>simple</ObjectComposition>
    <StructuralType>
      <Name></Name>
      <Extension></Extension>
    </StructuralType>
    <HardwareEnvironment>x86</HardwareEnvironment>
    <SoftwareEnvironment>OS: Windows 8.1 6.3, JVM:Oracle Corporation
1.8.0_31</SoftwareEnvironment>
    <InstallationRequirements></InstallationRequirements>
    <AccessInhibitors></AccessInhibitors>
    <AccessFacilitators></AccessFacilitators>
    <Quirks></Quirks>
    <MetadataRecordCreator>admin</MetadataRecordCreator>
    <MetadataCreationDate locale="CEST">
      <Date format="yyyyMMdd">20150501</Date>
      <Time format="HHmmssSSS">145653418</Time>
    </MetadataCreationDate>
    <Comments></Comments>
    <Files>
      <File
xmlns:nz_govt_natlib_xsl_XSLTFunctions="nz.govt.natlib.xsl.XSLTFunctions">
<FileIdentifier/>
<Path>C:\Users\Vclav\Desktop\Bakalsk_prce\Plohy\Vstupn_soubory\PHP\Mal_mnos
tv_dat\01simple.xlsx</Path>
<Filename>
<Name>01simple.xlsx</Name>
<Extension>xlsx</Extension>
</Filename>
<Size>6237</Size>
<FileDateTime>
<Date format="yyyyMMdd">20150407</Date>
<Time format="HHmmssSSS">232428640</Time>
</FileDateTime>
<Mimetype>application/open-office-1.x</Mimetype>
<FileFormat>
<Format>Open Office, </Format>
<Version/>
</FileFormat>
<Text>
<CharacterSet>ISO-8859-1</CharacterSet>
<MarkupLanguage/>
</Text>
</File>
</Files></Object>
```

4. Ukázkový příklad metadat MS Word souboru zapsaných v jazyce XML

```
<Object>
    <Name>new simple</Name>
    <ID>0</ID>
    <ReferenceNumber></ReferenceNumber>
    <GroupIdentifier></GroupIdentifier>
    <PersistentIdentifier></PersistentIdentifier>
    <MasterCreationDate locale="CEST">
      <Date format="yyyyMMdd">20150501</Date>
      <Time format="HHmmssSSS">145653546</Time>
    </MasterCreationDate>
    <ObjectComposition>simple</ObjectComposition>
    <StructuralType>
      <Name></Name>
      <Extension></Extension>
    </StructuralType>
    <HardwareEnvironment>x86</HardwareEnvironment>
    <SoftwareEnvironment>OS: Windows 8.1 6.3, JVM:Oracle Corporation
1.8.0_31</SoftwareEnvironment>
    <InstallationRequirements></InstallationRequirements>
    <AccessInhibitors></AccessInhibitors>
    <AccessFacilitators></AccessFacilitators>
    <Quirks></Quirks>
    <MetadataRecordCreator>admin</MetadataRecordCreator>
    <MetadataCreationDate locale="CEST">
      <Date format="yyyyMMdd">20150501</Date>
      <Time format="HHmmssSSS">145653547</Time>
    </MetadataCreationDate>
    <Comments></Comments>
    <Files>
      <File
xmlns:nz_govt_natlib_xsl_XSLTFunctions="nz.govt.natlib.xsl.XSLTFunctions">
<FileIdentifier/>
<Path>C:\Users\Vclav\Desktop\Bakalsk_prce\Plohy\Vstupn_soubory\MySQL\Mal_mn
ostv_dat\Adresa.doc</Path>
<Filename>
<Name>Adresa.doc</Name>
<Extension>doc</Extension>
</Filename>
<Size>3000</Size>
<FileDateTime>
<Date format="yyyyMMdd">20150126</Date>
<Time format="HHmmssSSS">145057541</Time>
</FileDateTime>
<Mimetype>text/html</Mimetype>
<FileFormat>
<Format>Hypertext Markup Language (HTML)</Format>
<Version>1.0</Version>
</FileFormat>
<Text>
<CharacterSet/>
<MarkupLanguage/>
</Text>
</File>
</Files></Object>
```

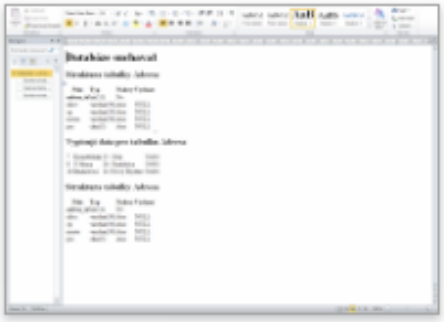
5. Ukázkový příklad metadat PDF souboru zapsaných v jazyce XML

```
<Object>
    <Name>new simple</Name>
    <ID>0</ID>
    <ReferenceNumber></ReferenceNumber>
    <GroupIdentifier></GroupIdentifier>
    <PersistentIdentifier></PersistentIdentifier>
    <MasterCreationDate locale="CEST">
      <Date format="yyyyMMdd">20150501</Date>
      <Time format="HHmmssSSS">145653678</Time>
    </MasterCreationDate>
    <ObjectComposition>simple</ObjectComposition>
    <StructuralType>
      <Name></Name>
      <Extension></Extension>
    </StructuralType>
    <HardwareEnvironment>x86</HardwareEnvironment>
    <SoftwareEnvironment>OS: Windows 8.1 6.3, JVM:Oracle Corporation
1.8.0_31</SoftwareEnvironment>
    <InstallationRequirements></InstallationRequirements>
    <AccessInhibitors></AccessInhibitors>
    <AccessFacilitators></AccessFacilitators>
    <Quirks></Quirks>
    <MetadataRecordCreator>admin</MetadataRecordCreator>
    <MetadataCreationDate locale="CEST">
      <Date format="yyyyMMdd">20150501</Date>
      <Time format="HHmmssSSS">145653678</Time>
    </MetadataCreationDate>
    <Comments></Comments>
    <Files>
      <File
xmlns:nz_govt_natlib_xsl_XSLTFunctions="nz.govt.natlib.xsl.XSLTFunctions">
<FileIdentifier/>
<Path>C:\Users\Vclav\Desktop\Bakalsk_prce\Plohy\Vstupn_soubory\MySQL\Mal_mn
ostv_dat\Adresa.pdf</Path>
<Filename>
<Name>Adresa.pdf</Name>
<Extension>pdf</Extension>
</Filename>
<Size>183160</Size>
<FileDateTime>
<Date format="yyyyMMdd">20150125</Date>
<Time format="HHmmssSSS">152705355</Time>
</FileDateTime>
<Mimetype>application/pdf</Mimetype>
<FileFormat>
<Format>Abobe PDF</Format>
<Version>1.7</Version>
</FileFormat>
<Text>
<CharacterSet>ISO-8859-1</CharacterSet>
<MarkupLanguage>unknown</MarkupLanguage>
</Text>
</File>
</Files></Object>
```

6. Ukázkový příklad metadat souboru XML zapsaných v jazyce XML

```
<Object>
    <Name>new simple</Name>
    <ID>0</ID>
    <ReferenceNumber></ReferenceNumber>
    <GroupIdentifier></GroupIdentifier>
    <PersistentIdentifier></PersistentIdentifier>
    <MasterCreationDate locale="CEST">
      <Date format="yyyyMMdd">20150501</Date>
      <Time format="HHmmssSSS">145654156</Time>
    </MasterCreationDate>
    <ObjectComposition>simple</ObjectComposition>
    <StructuralType>
      <Name></Name>
      <Extension></Extension>
    </StructuralType>
    <HardwareEnvironment>x86</HardwareEnvironment>
    <SoftwareEnvironment>OS: Windows 8.1 6.3, JVM:Oracle Corporation
1.8.0_31</SoftwareEnvironment>
    <InstallationRequirements></InstallationRequirements>
    <AccessInhibitors></AccessInhibitors>
    <AccessFacilitators></AccessFacilitators>
    <Quirks></Quirks>
    <MetadataRecordCreator>admin</MetadataRecordCreator>
    <MetadataCreationDate locale="CEST">
      <Date format="yyyyMMdd">20150501</Date>
      <Time format="HHmmssSSS">145654157</Time>
    </MetadataCreationDate>
    <Comments></Comments>
    <Files>
      <File
xmlns:nz_govt_natlib_xsl_XSLTFunctions="nz.govt.natlib.xsl.XSLTFunctions">
<FileIdentifier/>
<Path>C:\Users\Vclav\Desktop\Bakalsk_prce\Plohy\Vstupn_soubory\MySQL\Mal_mn
ostv_dat\Adresa.xml</Path>
<Filename>
<Name>Adresa.xml</Name>
<Extension>xml</Extension>
</Filename>
<Size>2176</Size>
<FileDateTime>
<Date format="yyyyMMdd">20150124</Date>
<Time format="HHmmssSSS">165822251</Time>
</FileDateTime>
<Mimetype>application/xml</Mimetype>
<FileFormat>
<Format>XML</Format>
<Version>1.0</Version>
</FileFormat>
<Text>
<CharacterSet>utf-8</CharacterSet>
<MarkupLanguage>XML</MarkupLanguage>
</Text>
</File>
</Files></Object>
```

7. Ukázka metadat Word Dokumentu z prostředí Microsoft Word 2010



Vlastnosti

Velikost	2,92 kB
Stránky	1
Slova	76
Celková doba úprav	0 minut
Název	Přidat název
Značky	Přidat značku
Komentáře	Přidat komentáře
Šablona	Normal.dotm
Stav	Přidat text
Kategorie	Přidat kategorii
Předmět	Zadat předmět
Základ hypertextového odkazu	Přidat text
Společnost	Zadat společnost

Související data

Naposledy upraveno	Nikdy
Vytvořeno	Dnes, 21:05
Naposledy vytištěno	Nikdy

Související uživatelé

Nadřízený	Zadat správce
Autor	Václav Suchánek Přidat autora
Autor poslední změny	Zatím neuloženo

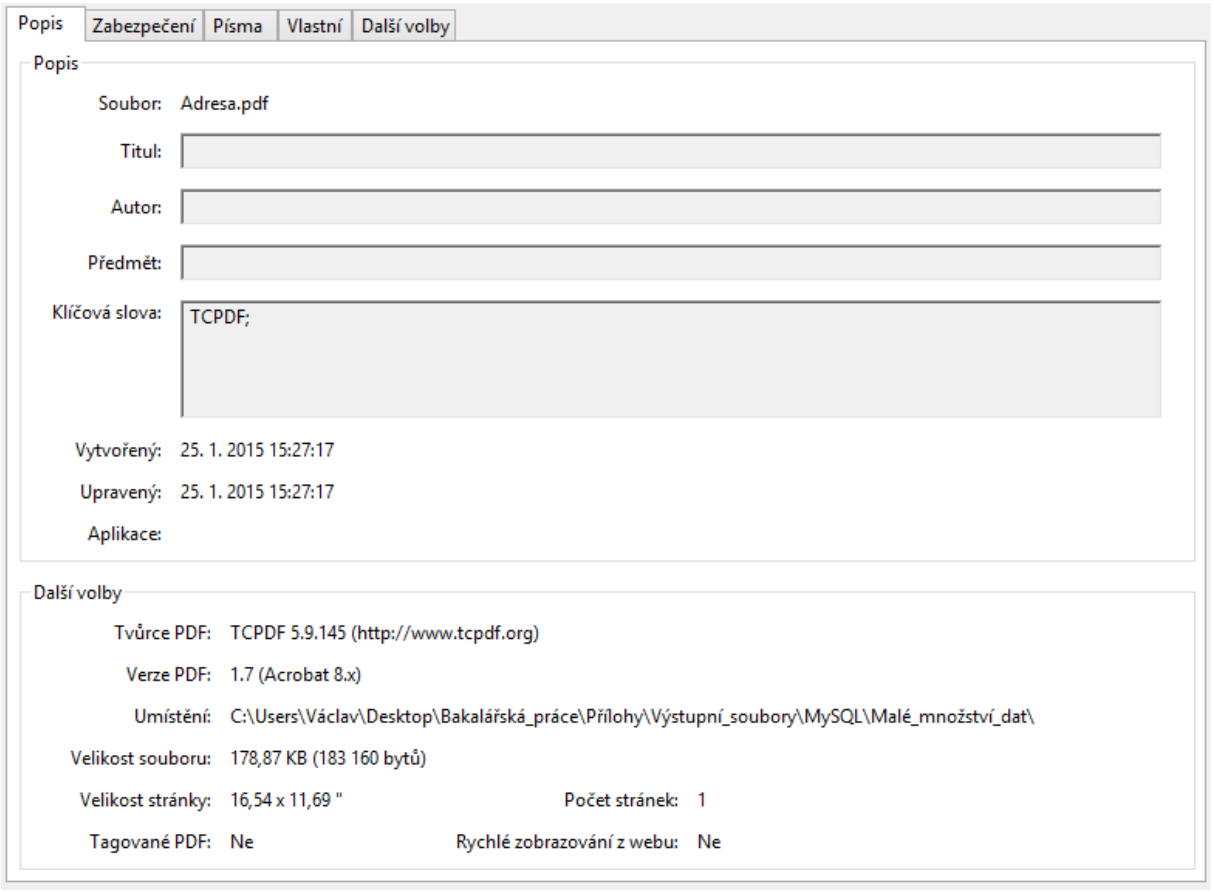
Související dokumenty

Otevřít umístění souboru

Zobrazit méně vlastností

Obrázek 4: Ukázka metadat souboru MS Word v prostředí Microsoft Word [vlastní zpracování]

8. Ukázka metadat souboru PDF z prostředí Adobe Reader XI



Obrázek 5: Ukázka metadat souboru PDF v prostředí Adobe Reader XI

9. Ukázkové příklady skriptů jazyků PHP a Java a souboru pro podporu výpočtů a grafů

- Přílohy.zip
  - Skripty_pro_export_dat
    - JavaExport
      - Pro_malé_množství_dat
        - JavaToCSV
        - JavaToExcel
        - JavaToPDF
        - JavaToWord
        - JavaToXML
      - Pro_velké_množství_dat
        - JavaToCSV
        - JavaToExcel
        - JavaToPDF
        - JavaToWord
        - JavaToXML
    - PHPExport
      - Pro_malé_množství_dat
        - PHP_to_PDF_and_Excel
        - PHP_to_Word_and_CSV
        - PHP_to_XML
      - Pro_velké_množství_dat
        - PHP_to_PDF_and_Excel
        - PHP_to_Word_and_CSV
        - PHP_to_XML
  - Výstupní_soubory
    - Java
      - Malé_množství_dat
      - Velké_množství_dat
    - MySQL
      - Malé_množství_dat

        - Velké_množství_dat
    - PHP
        - Malé_množství_dat
        - Velké_množství_dat
- Výpočty.xlsx

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: **Václav Suchánek**

Obor studia: Informační management (3)
Jméno a příjmení vedoucího práce: **Monika Borkovcová**

Název práce:
Výstupní formáty souborů dat z databáze a jejich použití v různých programovacích jazycích

Název práce v AJ:
Output file formats of data and its use in various programming languages

Podtitul práce:
Formáty exportovaných souborů a metody exportu

Podtitul práce v AJ:
Exported files and export methods

Cíl práce: Analýza různých formátů souborů pro export dat z databáze, popis metod generování dat za pomoci programovacích jazyků a vyhodnocení postupů a metodik pro navrženou aplikaci.

Osnova práce:
1. Úvod
2. Cíl a metodika práce
3. Teoretická východiska
4. Stanovení charakteristik výstupních formátů dat z databáze a 2 programovacích jazyků
5. Hodnocení charakteristik na základě vlastní tvorby
6. Vyhodnocení optimálních formátů dat pro navrženou aplikaci a stanovení doporučení pro použití daných jazyků
7. Výsledky
8. Závěry a doporučení
9. Literární zdroje

Projednáno dne: 16.10.2014

Podpis studenta

Podpis vedoucího práce