Technická univerzita v Liberci
Ekonomická fakulta

Studijní program: M6209 Systémové inženýrství a informatika
Studijní obor: Manažerská informatika

Využití XML v podnikové praxi

Usage of XML in Enterprises

DP-MI-KIN-2010-08
PETR NOVOTNÝ

Vedoucí práce: Ing. Vladimíra Zádová, Ph.D., Katedra informatiky
Konzultant: RNDr. Libor Bednařík, Tarmac CZ a.s.

Počet stran 71 Počet příloh 0

Datum odevzdání: 5. ledna 2010

Zadání

Prohlášení

Byl jsem seznámen s tím, že na mou diplomovou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé diplomové práce pro vnitřní potřebu TUL.

Užiji-li diplomovou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Diplomovou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím diplomové práce a konzultantem.

V Liberci, 5.1.2010

Podpis:

Anotace

Cílem této práce je nabídnout přehled možností aplikací jazyka XML a zaměřit se na praktické aspekty využití jazyka XML v podnikové praxi, ukázat možnosti využití XML v podnikové praxi a demonstrovat vhodné a nevhodné způsoby nakládání s XML. Dále je hledána odpověď na základní otázku, a to jakou formu XML dokumentů zvolit pro datově orientované dokumenty a jak jsou jednotlivé formy vhodné pro použití v databázovém systému Microsoft SQL Server 2008. Struktura práce je zvolena následovně: nabízí úvod do jazyka XML, představuje jednotlivé aplikace a jejich praktické využití, popisuje možnosti práce s XML, které nabízí Microsoft Office 2007 a detailně se věnuje problematice efektivnosti datových operací s XML soubory.

Klíčová slova: XML, databáze, efektivita, podniky

Annotation

The goal of this thesis is to offer a list of possible applications of the XML language and focus on practical aspects of usage of XML in enterprises, to show possibilities of using XML in enterprises and to demonstrate suitable and unsuitable ways of working with XML. Thesis is offering an answer to the question what form of XML documents to choose and how are individual solutions suitable for usage on database system Microsoft SQL Server 2008. The structure of thesis is following: offering an introduction to the XML, introducing applications of XML and their practical utilization, description of possibilities related to XML and offered by Microsoft Office 2007 and detailed description of effectives of data operations with XML files.

Keywords: XML, databases, effectiveness, enterprises

Obsah

Seznam zkratek a symbolů

BLOB	Binary Large Object
CLOB	Character Large Object
CSS	Cascade Style Sheets
DTD	Data Type Definition
GNU GPL	GNU General Public License
ISDS	Informační systém datových schránek
JDK	Java Development Kit
KML	Keyhole Markup Language
MSDN	Microsoft Developer Network
MVČR	Ministerstvo vnitra České republiky
SEO	Search Engine Optimization
SGML	Standard Generalized Markup Language
WAP	Wireless Application Protocol
WWW	World Wide Web
XHTML	Extensible HyperText Markup Language
XHTML MP	Extensible HyperText Markup Language Mobile Profile
XML	Extensible Markup Language
XSL	Extensible Stylesheet Language

Seznam Tabulek

Seznam obrázků

Seznam grafů

1 Úvod

Od počátku existence výpočetní techniky se vyskytl bezpočet systémů ukládající data specificky pro každého výrobce s absencí standardizace datových formátů, tudíž i s nepřenositelnými výstupy a neumožňující vstupy z konkurenčních systémů. Tento trend stále do jisté míry přetrvává, ale již po určitou dobu dochází mezi výrobci softwaru k podpoře vybraných způsobů záznamu dat, mezi něž se řadí i XML. Cílem této práce je zaměřit se na praktické možnosti využití XML v podnikové praxi.

Tato práce vznikla proto, že řada publikací a manuálů uvádí, že popisovaný software je schopen pracovat s XML, detailní seznámení s XML a co to je XML, popis jazyka, hierarchie, DTD, schéma, transformace, XQuery, vyjmenování možných úkonů pro daný software. Častým jevem je uvádění údajů o syntaxi jazyka XML, a absence návodů, jak s XML zacházet efektivně a čeho se vyvarovat. Jmenovitě [19] zmiňuje možnost zápisu dat pomocí atributů a elementů, předkládá argumenty pro jednotlivé možnosti, ale již nezmiňuje, jak popisované způsoby dat ovlivňují chod SQL Serveru, nebo [18] popisuje, jak importovat XML data, ale již nesdělí problémy, které mohou nastat.

Práce je rozdělena na teoretickou a praktickou část. V teoretické části je čtenář seznámen s jazykem XML a z něj odvozenými aplikacemi. V praktické části jsou představeny výsledky experimentů s vyvozenými závěry. Praktická část je zaměřena na datově orientované dokumenty a problematiku spojenou s jejich přenosem.

Následující kapitola je věnována vysvětlení zákonitostí jazyka XML, způsobům ukládání dat, DTD a XML schématu. Třetí kapitola pojednává o XML technologiích a konkrétních aplikacích pro podnikovou praxi. Čtvrtá kapitola nabízí přehled možností využití XML v aplikacích Office a OpenOffice a zvláštnosti spojené s jednotlivými programy. Pátá kapitola obsahuje výsledky a závěry plynoucí z pokusů s datově orientovanými dokumenty na SQL Serveru 2008 a nativní XML databázi eXist.

2 Syntaxe XML

Pro poskytnutí základního přehledu o jazyce XML následuje popis vybraných prvků syntaxe. XML je značkovací jazyk odvozený z jazyka SGML. XML dokument se skládá z deklarace, odkazu na XML schéma nebo DTD, jednoho kořenového elementu a jednoho nebo více elementů zařazených do kořenového elementu. Mezi jednotlivými elementy existuje hierarchické uspořádání typu rodič – dítě, tzn. element <*objednavky*> obsahuje dílčí elementy <*objednavka*> a každý z elementů <*objednavka*> má své vlastní elementy obsahující samotná data, viz obrázek 1. Systém odpovídá logickému uspořádání, kdy v šanonu objednávky bude více dílčích objednávek a každá bude obsahovat specifické údaje.

```
<?xml version="1.0" encoding="windows-1250"?>
<Objednavky>
        <objednavka>
                <jmeno>Jan Novák</jmeno>
                <telefon>728964285</telefon>
                <mnozstvi>50</mnozstvi>
                <cena>5000</cena>
                 <zbozi>004</zbozi>
        </objednavka>
        <objednavka>
                <jmeno>Jiří Vokurka</jmeno>
                <telefon>728970254</telefon>
                <mnozstvi>10</mnozstvi>
                <cena>8000</cena>
                <zbozi>039</zbozi>
         </objednavka>
</Objednavky>
```

Obr. 1 Syntaxe XML

Zdroj: vlastní

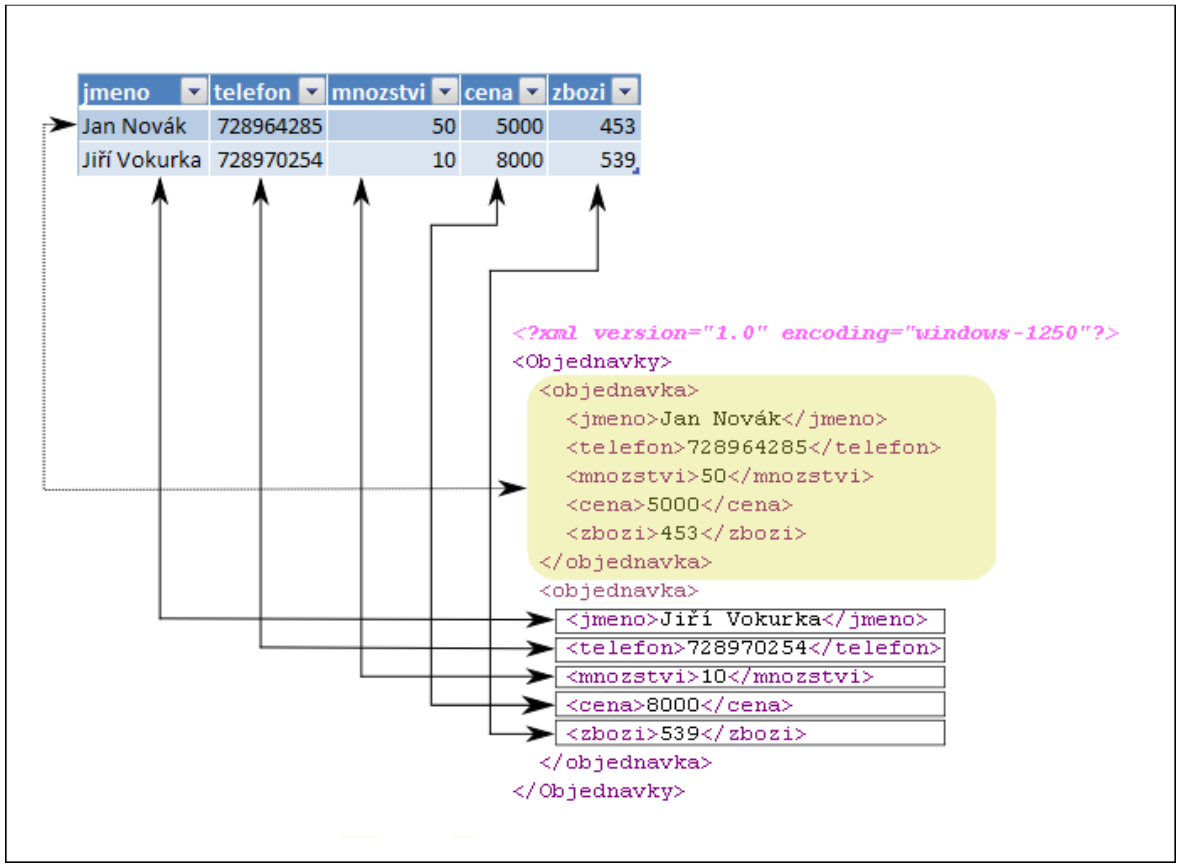
XML nabízí uložení dvou základních typů dokumentů: datově a dokumentově orientované dokumenty.

2.1 Datově orientované dokumenty

Vyznačují se pravidelnou strukturou, komplexní hierarchií, vysokou granularitou, z pravidla jsou určeny pro strojové zpracování, obvykle jsou také strojově vytvořeny. XML slouží pouze jako přenosné médium. Příkladem datově orientovaného dokumentu je kód na Obr. 1.

Ukládání dat do datově orientovaného dokumentu je možné dvěma způsoby, a to tak, že se data vyskytují jako elementy, nebo atributy. Oba přístupy jsou předvedeny na příkladu dat obsažených v obrázku 1 - objednávka obsahující údaje o dvou osobách.

2.1.1 Uložení dat jako elementy

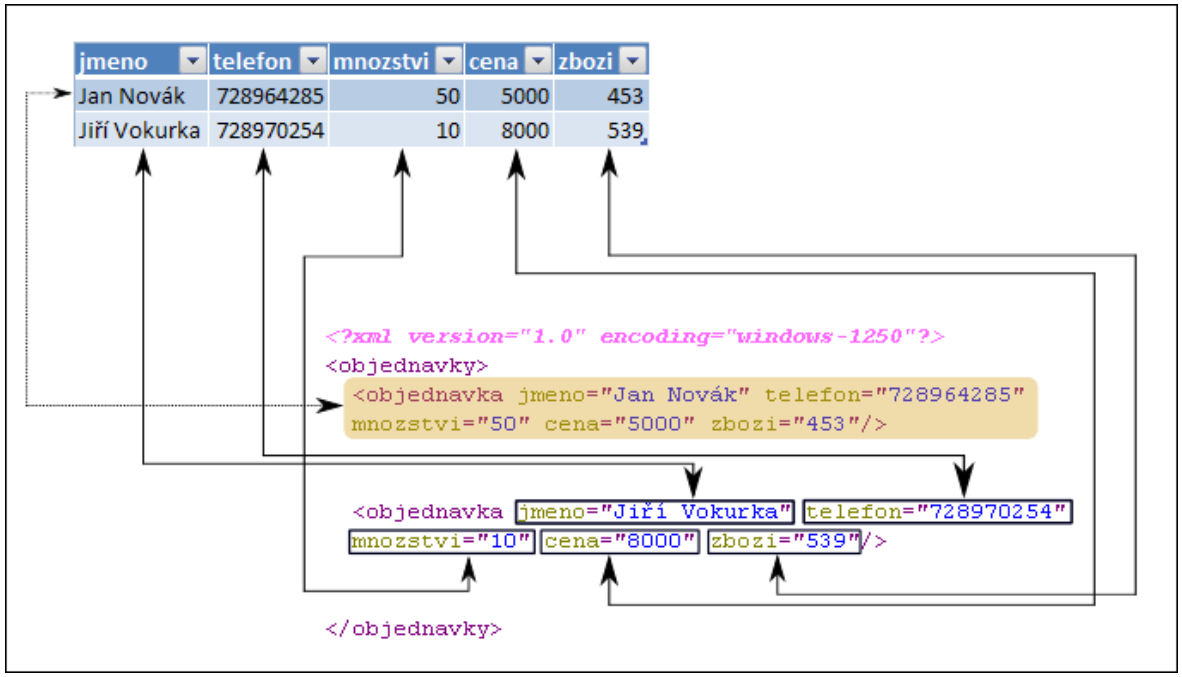


Obr. 2 Zápis dat formou elementů

Zdroj: vlastní

Požadovaná data jsou uložena jako elementy, které v relačním modelu reprezentují jednotlivé relace. Při této formě zápisu záleží na hierarchii jednotlivých elementů. Vztah mezi jednotlivými elementy a buňkami tabulky je demonstrován na obrázku 2. Tento přístup je mimo jiné vykoupen vyšší velikostí souborů způsobenou přítomností počátečních a koncových značek u každého prvku.

2.1.2 Uložení dat jako atributy



Obr. 3 Zápis dat formou atributů

Zdroj: vlastní

Při využití této formy zápisu atributy odpovídají sloupcům v databázovém relačním modelu, v jednom elementu se nesmí vyskytovat dva atributy se stejným názvem. Stejně jako u zápisu pomocí elementů, záleží na pořadí atributů v prvním řádku. Vztah mezi buňkami v tabulce, elementy a atributy je demonstrován na obrázku 3.

2.2 Dokumentově orientované dokumenty

Jsou pravým opakem datově orientovaných dokumentů. Mají nepravidelnou strukturu, jsou určeny pro čtení lidmi, při porovnání s datově orientovanými dokumenty jsou málo členité, vyhledávání v takových dokumentech je omezeno na fulltextové dotazy. Často se jedná o písemnosti publikované prostřednictvím XML s využitím transformací textu a grafickými efekty. Příkladem takového dokumentu je úryvek z této práce, viz obrázek 4.

```
<?xml version="1.0" encoding="windows-1250"?>
<DP>
<kapitola1>
  <nadpis1>1. Úvod</nadpis1>
<odstavec1> Od počátku existence výpočetní techniky se vyskytl bezpočet systémů ukládající data specificky pro každého výrobce s absencí standardizace datových formátů, tudíž i s nepřenositelnými výstupy a neumožňující vstupy z konkurenčních systémů. Tento trend stále do jisté míry přetrvává, ale již po určitou dobu dochází mezi výrobci softwaru k podpoře vybraných způsobů záznamu dat, mezi něž se řadí i XML. Cílem této práce je zaměřit se na praktické možnosti využití XML v podnikové praxi.
</odstavec1>
   <odstavec2>.
   .
   .
   </odstavec2>
 </kapitola1>
```

Obr. 4 Dokumentově orientovaný dokument

Zdroj: vlastní

2.3 DTD

Definice typu dokumentu umožňuje deklarovat obsah dokumentu. DTD může být součástí dokumentu, nebo uloženo ve vlastním souboru, poté je nutné doplnit dokument obsahující data odkazem na tento soubor. DTD umožňuje kontrolovat obsah dokumentu podle zvolených kritérií - deklarace elementů, atributů, entit a notací.

Elementy se dají deklarovat tak, že budou prázdné, obsahovat jakoukoliv hodnotu, počet hodnot ≥ 0, nebo text. Atributy lze deklarovat tak, že mohou obsahovat libovolný textový řetězec, nebo slova skládající se z písmen a čísel, nebo výčtem přípustných hodnot. Dále je možné deklarovat opakující se elementy pomocí parametrických entit.

DTD je vhodné použít, pokud je třeba zajistit kontrolu obsahu elementů jmenným výčtem povolených hodnot, nebo za situace, kdy si je tvůrce jistý, že XML dokument nebude nikdy v budoucnu zpracován v databázi – pro tuto potřebu je DTD nedostačující. Využití DTD ztrácí význam za situace, kdy existuje možnost pokrytí stávajících a možných budoucích potřeb prostřednictvím XML schématu, které je schopné splnit stejnou úlohu jako DTD a posloužit za situace, kdy bude třeba dokument zpracovat v databázi. Příklad DTD je na obrázku 5.

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT Objednavky ((objednavka+))>
<!ELEMENT objednavka ((jmeno, telefon, mnozstvi, cena, zbozi))>
<!ELEMENT jmeno (#PCDATA)>
<!ELEMENT telefon (#PCDATA)>
<!ELEMENT mnozstvi (#PCDATA)>
<!ELEMENT cena (#PCDATA)>
<!ELEMENT zbozi (#PCDATA)>
```

Obr. 5 DTD k příkladu z obrázku 1

Zdroj: vlastní

2.4 XML Schéma

XML schéma je vyšší formou DTD, která vznikla kvůli používání XML v databázích, které kladou vyšší nároky na definici dat, než umožňuje DTD. Schéma není povinnou součástí XML, jeho přítomnost není na závadu, ale řada moderních databází si poradí s XML dokumentem i bez něj. Přestože XML umožňuje rozmanité specifikace pro definování datových typů, různé databázové systémy mohou mít problémy s detailně definovanými datovými typy. Pokud víme, že budeme XML využívat pouze k přenosu dat

a použijeme schéma, není vhodné zacházet do příliš specifikovaných datových typů a je výhodnější omezit se na základní. Pokud se nelze vyhnout problémům při datových operacích se znaky, které neodpovídají standardu Unicode, řešení takových problémů obvykle spočívá v konverzi non-Unicode znaků na Unicode znaky. K tomu lze využít například Microsoft Visual Studio.

Na obrázku 6 je schéma vytvoření k příkladu z obrázku 1, které obecně popisuje jednotlivé elementy.

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:element name="zamestnanec">
  <xs:complexType>
   <xs:sequence>
    <xs:element name="jmeno" type="xs:string"/>
    <xs:element name="telefon" type="xs:"string/>
    <xs:element name="mnozstvi" type="xs:integer"/>
    <xs:element name="cena" type="xs:integer"/>
    <xs:element name="zbozi" type="xs:integer"/>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
</xs:schema>
```

Obr. 6 Schéma k příkladu z obrázku 1

Zdroj: vlastní

Schéma ke stejnému příkladu lze vytvořit i naprosto odlišným způsobem a to explicitním popsáním jednotlivých elementů, viz obrázek 7. Takovýto přístup k tvorbě schématu se dá využít ke kontrole entitní integrity pomocí porovnání obsahu XML souboru se schématem. Nevýhodou tohoto přístupu je nemožnost tvorby schématu ručně pro velké soubory.

```
?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
	<xs:simpleType name="T_zbozi">
		<xs:restriction base="xs:byte">
			<xs:enumeration value="39"/>
			<xs:enumeration value="4"/>
		</xs:restriction>
	</xs:simpleType>
	<xs:simpleType name="T_telefon">
		<xs:restriction base="xs:int">
			<xs:enumeration value="728964285"/>
			<xs:enumeration value="728970254"/>
		</xs:restriction>
	</xs:simpleType>
	<xs:simpleType name="T_mnozstvi">
		<xs:restriction base="xs:byte">
			<xs:enumeration value="10"/>
			<xs:enumeration value="50"/>
		</xs:restriction>
	</xs:simpleType>
	<xs:simpleType name="T_jmeno">
		<xs:restriction base="xs:string">
			<xs:enumeration value="Jan Novák"/>
			<xs:enumeration value="Jiří Vokurka"/>
		</xs:restriction>
	</xs:simpleType>
	.
	.
	.
```

Obr. 7 Explicitní schéma vygenerované programem Altova XMLSpy

Zdroj: vlastní

3 Aplikace XML

Tato kapitola je rozdělena do dvou podkapitol. První předkládá přehled technologií založených na XML, nebo s ním souvisejících, druhá je zaměřena na konkrétní aplikace jazyka XML.

3.1 Technologie pracující s XML

XML je jazyk pro tvorbu dalších jazyků, jeho aplikací je právě tolik, kolik jich uživatelé vytvoří. V následujících kapitolách budou zmíněny nejznámější a nejvíce používané aplikace XML.

3.1.1 XHTML

Jazyk XHML se řadí do skupiny jazyků, jejichž předchůdcem je jazyk SGML. Jazyk HTML byl původně stvořen k výměně vědeckých a technických dokumentů, ale vlivem nepředvídaných událostí se vyvinul za hranice očekávání a stal se aplikací XML používanou pro tvorbu WWW.

XHTML vznikl přeformulováním HTML jako XML aplikace [13] a je navržen tak, aby ho dokázalo přečíst co největší množství prohlížečů. První verze následovala po HTML 4 a oproti HTML přinesla omezení vyplývající z pravidel jazyka XML. Zásadní změnou byl požadavek na správnou formulaci dokumentu: všechny elementy musí být uzavřeny, vnořené elementy musí být uzavírány v odpovídajícím pořadí, rozpoznávání velkých a malých písmen ve značkách, neprázdné elementy musí být uzavřeny příslušnou koncovou značkou, obsah atributů musí být v uvozovkách, prázdné elementy musí mít koncovou značku, nebo první značka musí být zakončena lomítkem, následovaným

uzavírací značkou a další specifikace [13]. Vývoj standardů XHTML proběhl do verze 2.0, kde byl zastaven.

Poslední verzí standardu, stále ve vývoji, pro tvorbu internetových stránek s použitím XML je HTML 5.0. Oproti předcházejícím verzím HTML 4 a XHTML 1, které jsou vzájemně neslučitelné, přináší sjednocení obou jazyků - HTML 5 umožňuje použití HTML a XML syntaxe.

Klíčovým kritériem vypovídajícím o webu, je rychlost s jakou se načítá, jak rychle provádí požadované operace. Pokud budou k vytváření výstupů pro uživatele používány XML soubory, je nutno věnovat zvýšenou pozornost problematice volby parseru. Různé parsery dosahují odlišných výsledků pro soubory rozličných velikostí [12].

3.1.2 Mobilní internet

Pojmem mobilní internet (WAP) se označuje přístup ke stránkám pomocí přenosných zařízení bezdrátovým spojením. Internet, ať už je zprostředkováván jakoukoliv technologií, je úspěšný ve světě „pevného“ připojení, ale WAP v České republice stále nedosáhl potenciálu dosaženého ve vyspělých zemích. Za technické důvody nezdaru se dá označit velikost displejů mobilních zařízení, různé velikosti displejů, omezená kapacita paměti pro soubory cookies, šířka pásma a způsoby navigace na stránkách specifické pro každé zařízení. Všechny technické překážky se promítají do potřeby napsat stránky tak, aby mohly být prohlíženy jak z mobilních zařízení, tak ze zařízení s pevným připojením, eventuelně mít k dispozici možnost volby mezi stránkami pro mobilní prohlížeče a zařízení s pevným připojením. Současný standard pro mobilní internet WAP 2.0 používá variaci jazyka XML speciálně upravenou pro mobilní telefony, XHTML MP.

3.1.3 Sledování osob

Jonathan F. Spencer uvádí ve své publikaci [2] zajímavý příklad aplikace XML. Stručně popisuje, jak využít XHTML ke sledování aktivit jedinců na internetových fórech. Podstata spočívá v získání přístupu do skupiny, získání kopie diskuse transformací HTML kódu do potřebné formy a následná analýza. Původní myšlenku lze rozšířit na komunitní weby (Facebook, MySpace), které se velmi snadno mohou stát nástrojem ke sledování aktivit vybraných osob. Největší překážkou zůstává získání přístupu k cílové skupině uživatelů, samotná realizace stalkerské aplikace je již „pouhou“ záležitostí naprogramování robota a podmínek, za jakých bude prováděna analýza aktivit.

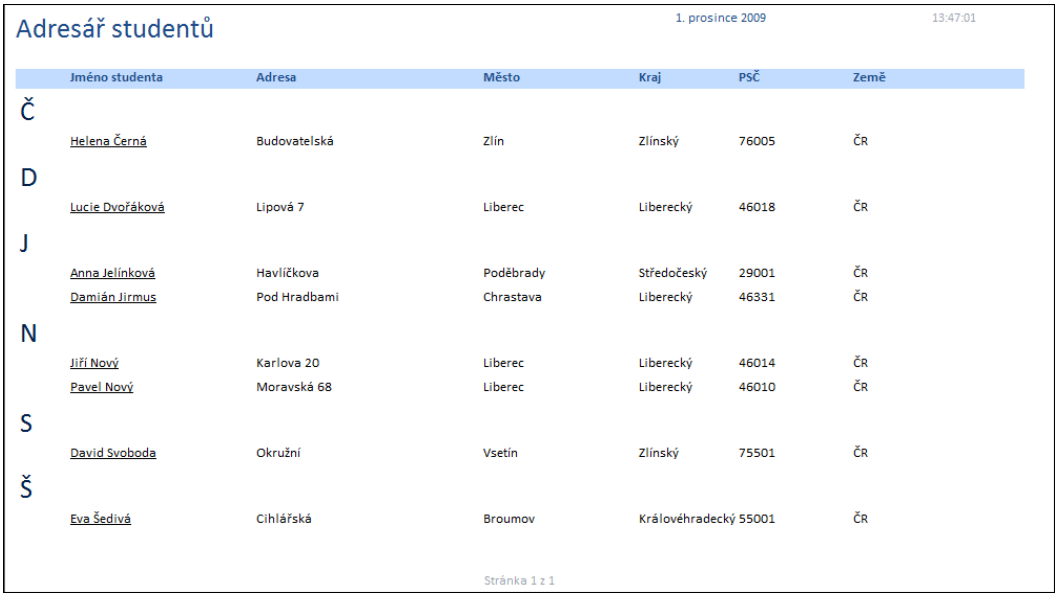
Ať je morální pohled na takové počínání jakýkoliv, sledování kohokoliv v kyberprostoru není složitou záležitostí. Stačí přečíst kód příslušné stránky a dále s ním nakládat dle vlastních požadavků. Pro příklad: zaměstnavatel chce sledovat své zaměstnance, kteří mají přístup k citlivým údajům, nebo jej zajímá, zdali se zaměstnanci věnují v pracovní době pouze práci. K tomu lze využít technologie, použité k vytvoření samotné internetové stránky, XHTML. Pozorovatel přitom nemusí sledovat své cíle v reálném čase, ale jednou za čas si prohlédne výpis aktivity, příspěvky, zveřejněné obrázky, filtry ho upozorní na předem zadané fráze, čas zveřejnění příspěvku.

3.2 Konkrétní využití XML

Tato podkapitola je zaměřena na prakticky použitelné výstupy z XML, produkty postavené na XML a konkrétní možnosti jak může podnik využít XML ke svým potřebám.

3.2.1 Prezentace

Jazyk XML je navržen způsobem, který odděluje informace obsažené v dokumentu od grafického vzhledu. Řada koncových uživatelů nemusí XML vůbec ovládat a bude očekávat zobrazení dat s grafickou úpravou, kde konkrétní podobu výstupu určí stylový jazyk. Použití jednoho stylu není vázáno na konkrétní dokument, lze mít řadu dokumentů se stejným stylem, nebo jeden dokument s více styly. Některé programy nabízí funkci exportu obsahu do XML s připojením stylů, nebo transformací například do jazyka HTML (Access 2007). Příkladem takto transformovaného dokumentu je výstup na obrázku 8. Další příklady jsou k nalezení na přiloženém nosiči v adresáři Měření.



Adresář studentů

1. prosince 2009 13:47:01

Jméno studenta	Adresa	Město	Kraj	PSČ	Země
Č					
Helena Černá	Budovatelská	Zlín	Zlínský	76005	ČR
D					
Lucie Dvořáková	Lipová 7	Liberec	Liberecký	46018	ČR
J					
Anna Jelínková	Havlíčkova	Poděbrady	Středočeský	29001	ČR
Damián Jirmus	Pod Hradbami	Chrastava	Liberecký	46331	ČR
N					
Jiří Nový	Karlova 20	Liberec	Liberecký	46014	ČR
Pavel Nový	Moravská 68	Liberec	Liberecký	46010	ČR
S					
David Svoboda	Okružní	Vsetín	Zlínský	75501	ČR
Š					
Eva Šedivá	Cihlářská	Broumov	Královéhradecký	55001	ČR

Stránka 1 z 1

Obr. 8 XML transformované do HTML

Zdroj: vlastní

3.2.2 Publikace textů

Základem pro správné používání jakéhokoliv zařízení, programu, nebo technologie je přístup k příslušné dokumentaci. Dnes se již nelze spokojit s jedním výstupem, ale je nutné zajistit přístupnost v nápovědě samotné aplikace, tištěnou formu, zveřejnění na internetu, distribuce na přenosná média. Nejefektivnějším řešením je získat potřebné formáty

z originálního a často se měnícího díla, nejlépe s minimálním úsilím. Existuje celá řada nástrojů nabízející řešení na míru, například encyklopedie, ale nevýhody takových systémů spočívají v nepružnosti a omezení pramenících z designu.

Univerzálního řešení lze dosáhnout pouze s použitím univerzální technologie, v tomto případě XML. Jazyk XML je orientován sémanticky, nikoliv vizuálně, vytvořený textový dokument lze dodatečně snadno naformátovat. K publikaci textů prostřednictvím XML jsou k dispozici dvě skupiny nástrojů, pracující na stejném principu. První jsou projekty s GNU GPL licencemi - jedním z takových nástrojů je DocBook. Druhou skupinu tvoří zpoplatněné programy pracující na stejném principu jako DocBook, které se spoléhají na neochotu lidí pracovat se samotným XML kódem, jmenovitě se jedná například o: Epic, XMetaL, epcEdit. Tvorba XML dokumentů v takových editorech se podobá psaní textu ve Wordu, nebo kterémkoliv jiném textovém editoru a to včetně vkládání obrázků a tabulek. Oproti Wordu zde existuje zásadní rozdíl: napsaný text je ukládán do datově orientovaného XML dokumentu.

Univerzálnost používání XML k psaní textů se projeví až při potřebě publikovat obsah ve více formátech. Samotný DocBook umožňuje naformátovat soubor pro výstupy: tištěné, HTML, JavaHelp, nápověda pro Eclipse [14].

3.2.3 E-learning

Digitální knihovna s multimediálním obsahem bez metadat se bude potýkat s absencí funkce vyhledávání obsahu. Doplňování metadat k textům ztratilo z pohledu vyhledání obsahu s příchodem internetových vyhledávačů svůj význam, ovšem stále hraje důležitou roli při SEO webu. Doplňování metadat o obsahu má stále svůj význam pro audiovizuální díla, která svou technologií zabraňují robotům v indexaci jejich obsahu (flash animace, java, video, atd.). Informace o tvůrci, názvu a datu zveřejnění jsou pro vyhledávání nedostatečné a pořizování přepisů mluveného slova je příliš nákladné. Ačkoliv model založený na vyplňování metadat tvůrci stránek obsahující text je nefunkční a to díky

stránkám s metadaty neodpovídajícím obsahu stránek, systém jako takový není k zavržení pro weby s multimediálním obsahem, respektive prozatím neexistuje jiný způsob k popsání obsahu. Pokud se rozhodneme pro popis pomocí metadat a XML, vyústí naše aktivita ve dvě vzájemně propojené části digitální knihovny: databázi dat a databázi metadat. V praxi lze využít nativní XML databáze k ukládání, fulltextovému a cílenému vyhledávání metadat a dokumentově orientovaných dokumentů v rámci podnikového informačního systému.

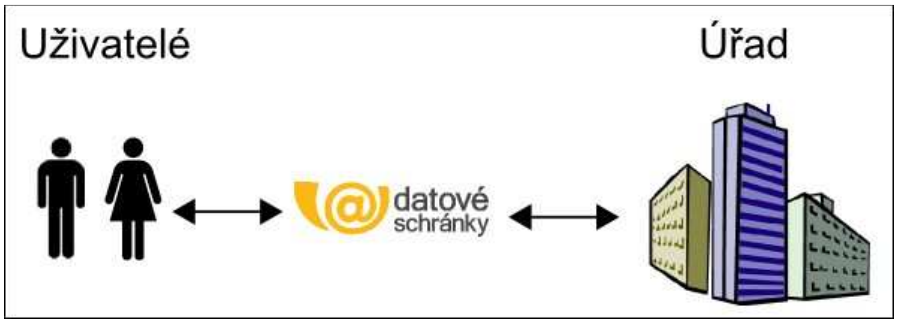
3.2.4 Výměna dat

V situaci, kdy každý výrobce software používá svůj formát pro ukládání dat, vyvstávají komplikace při jejich přenosu. Různé programy využívají různé, vzájemně nekompatibilní formáty a přenést data z jednoho programu do druhého, při využití vlastního datového formátu bývá mnohdy nemožné. Struktura, kterou používá XML, vybízí k využití jako univerzálního přenosového média. Je třeba vzít na zřetel to, že značkovací jazyky využívané k ukládání dat jsou mnohem méně úsporné, než datové formáty přímo navržené k ukládání dat. V praxi jsou využívány programy, které nabízí možnost uložení dat do XML (Microsoft Office 2007, Microsoft SQL Server 2008, Oracle, aj.).

3.2.4.1 Datové schránky

Dne 1. července 2009 došlo k nabytí účinnosti zákona č. 300/2008 Sb., o elektronických úkonech a autorizované konverzi dokumentů, který je ve znění pozdějších novel známý jako zákon o datových schránkách. Zákon udává povinnost orgánům veřejné moci a právnickým osobám zřízení datové schránky.

Datová schránka je forma autentizované elektronické komunikace s úřady nahrazující písemnou korespondenci ve styku se státní správou, viz obrázek 9.



Obr. 9 Schéma komunikace se státní správou pomocí datové schránky

Zdroj: vlastní

Vyhláška MVČR č. 194/2009 Sb. [16] mimo jiné specifikuje přípustné formáty pro zprávy doručované prostřednictvím systému datových schránek, XML je jedním z přípustných formátů.

Od 1. ledna 2010 bude možné přes datové schránky zasílat faktury a to mezi soukromými subjekty. Faktura může být vytvořena příslušným pohledem do databáze a uložením výstupu do XML s případným připojením transformace. Cena České pošty za obyčejné psaní do 50g je 10 Kč, 26 Kč za doporučené psaní do 50g, ceny jsou platné k 4. 12. 2009. Cena za datovou zprávu zaslanou systémem ISDS nebyla k 9. prosinci 2009 stanovena, nelze předložit detailní analýzu nákladů. Pokud by byla cena jedné datové zprávy rovna ceně obyčejného dopisu, 10 Kč, bylo by používáním ISDS k zasílání faktur dosaženo úspor rovným nákladům na obálky a času potřebnému k jejich odeslání. Systém ISDS funguje na principu „doporučené pošty“, eviduje doručení zpráv, což není u obyčejných dopisů možné. Při dodržení základních bezpečnostních pravidel lze zajistit přístup k datové schránce pouze pro pověřené osoby, čehož u poštovních schránek dosáhnout nelze.

3.2.5 Relační databáze

Relační databázové systémy představují efektivní nástroj ke správě dat. Problém vyvstává, pokud je třeba použít data, uložená v relační databázi, v softwaru nekompatibilním s datovým formátem relační databáze. Východiskem z takové situace je oddělit část databáze a uložit data v jazyce XML.

3.2.5.1 Relační model

Základem relačního modelu je relace – seskupení sloupců (atributů) a řádků. Data v relaci jsou ukládána po řádcích, záznamy v jednom řádku tvoří relaci. Správná integrita dat je zajištěna, mimo obvyklých definic datových typů, prostřednictvím domén, které přesně specifikují, jaká data mohou být vložena do zvoleného atributu, např. datum může mít tvar Den/Měsíc/Rok, nebo Měsíc/Den/Rok. Obojí reprezentuje stejné informace, ale v různých částech světa jsou k zápisu data využívány různé standardy a záznamy mohou být přečteny správně, ale existuje možnost přiřazení nesprávného významu. Data je možné normalizovat.

3.2.5.2 Normalizace

Je proces, který označuje uspořádání databáze do takové struktury, která eliminuje nekonzistence, redundanci dat a spoří místo na pevním disku při použití normálních forem, definovaných podmínek podoby databáze. Nejčastěji se provádí normalizace do třetí normální formy, nebo do Boyce-Coddovy normální formy.

3.2.5.3 Integrita

Integrita databáze zajišťuje správnost, platnost a konsistenci databáze. Integrita databáze se rozlišuje na doménovou, referenční a entitní.

Doménová integrita zajišťuje správnost hodnot atributů porovnáním s přípustnými hodnotami.

Entitní integrita udává, že každá relace musí mít primární klíč, který ji jednoznačně identifikuje.

Referenční integrita definuje vztah mezi dvěma tabulkami, mohou nastat dva stavy:

- 1:1 - záznam v jedné tabulce odpovídá právě jednomu záznamu v jiné tabulce, např. každé osobě z dané množiny je přiděleno unikátní identifikační číslo.
- 1:n - záznam v jedné tabulce odpovídá více záznamům v jiné tabulce, např. do třídy je zapsáno n žáků.

3.2.5.4 Integrace relační databáze a XML

K integraci XML do relační databáze existují tři přístupy:

První: uložení celého XML dokumentu jako textu v jednom databázovém atributu, CLOBu. Nevýhodami takovéhoto řešení je nemožnost uzavřít části dokumentu a nemožnost integrace s existujícím schématem databáze. Pro tento přístup k integraci XML a relační databáze je obvyklé použití rozšíření relační databáze o transakční manager vedoucí ke zlepšení výkonu při dotazování [11].

Druhý: rozložení XML dokumentu na dílčí části a jejich uložení v relační databázi. Tento přístup má nevýhody spočívající v nadměrných nárocích na join operace při dotazování a nemožnosti integrace s existujícím schématem databáze. Oproti předcházejícímu přístupu je výhodou možnost rozdělení dokumentu na části a pro stanovená pravidla rozdělení spojená se strukturou dokumentu, je dotazování na obsah snazší. První dva přístupy je vhodné zvolit, pokud potřebujeme uchovat integrovaná data pouze přechodně, budeme pracovat s oddělenými částmi. Pro trvalé uchování XML dokumentu je vhodné zvolit třetí přístup.

Třetí: mapování XML dokumentu podle schématu relační databáze a jeho následné uložení. Relační databáze podporující XML se potýkají s faktem, že relační model je zcela odlišný od XML modelu. XML obsah se do relační databáze dostává v několika krocích: načte se XML schéma, nebo si jej databáze sama vytvoří, a následně načte XML data.

Systém zpracuje XML schéma a vytvoří podle něj relace, do kterých následně zapíše data. Problém nastává, pokud proběhne dekompozice do normalizované databáze, která vyústí ve velké množství vzájemně propojených tabulek, což může být důsledkem odlišných návrhů databází, nebo odlišných datových modelů. Dotazování nad takovou databází vyžaduje mnoho join operací a představuje dodatečný požadavek na výkon při tvorbě pohledů [4,7].

Při mapování do nenormalizované databáze odpadá problém mnoha tabulek, ale může dojít k výskytu neatomických hodnot, funkčním a multifunkčním závislostem, což vede k redundanci v databázi [7].

3.2.6 Nativní XML databáze

Jejich model je speciálně navržen pro práci s XML daty a je založen na XML souborech a nad nimi postavené aplikaci, tudíž umožňují podobné funkce jako relační databáze (dotazování nad obsahem, aktualizace, vzdálený přístup, atd.), ale neexistuje všeobecně uznávaný standard datového formátu a každá nativní XML databáze používá svůj systém. Nejčastější využití nativních databází je k ukládání dokumentově orientovaných dokumentů a to kvůli dotazovacím jazykům, které umožňují snazší dotazování, než relační databáze vyžadující komplexní dotazy [15]. K dotazování se používají jazyky XPath a XQuery.

Ukládání dat v nativní XML databázi přináší některé výhody:

- rozlišení obsahu dokumentu od značek umožňující fulltextové vyhledávání XML obsahu
- na rozdíl od relačních databází nevyžadují schéma
- schéma lze měnit, aniž by se muselo zasahovat do samotných dat
- lze zaznamenávat data, aniž by měl tvůrce předem rozvržen vlastní obsah a definované datové typy

- dostupná rychlost při získávání dat z vhodných datových struktur. Komplikovaná tabulka (objednávka-zákazník-zboží-datum-čas) by byla v relační databázi uložena ve specifických tabulkách a při dotazování by bylo třeba spojeni těchto tabulek. Oproti tomu stojí takováto objednávka vyjádřená jako jeden XML soubor.
- vyšší rychlost dotazu pokud se dotazuje na XML dokument a ne na spojení s dalšími XML dokumenty. Toto může být do jisté míry omezením, zejména pokud je databáze nesprávně navrhnuta
- vhodné pro částečně strukturovaná data – v relační databázi by bylo plýtváno místem kvůli častému výskytu null hodnot, zatímco XML toto vyřeší vynecháním elementu.

Povaha nativních XML databází přináší také některé nevýhody:

- databáze je efektivní, pouze pokud dominuje jeden způsob zobrazování dat, tomu musí být uzpůsobena struktura dat
- nativní XML databáze většinou umožňují „pouze“ XML výstup. Pokud je požadavek na jiný formát, vyvstává problém.

Nativní XML databáze lze rozdělit do dvou skupin a s ohledem na způsob, s jakým přistupují k XML. Tyto skupiny jsou: produkty zaměřující se na práci s dokumenty a aplikace ke správě dat, přičemž častým problémem nativních XML databází bývá absence přesně stanoveného datového modelu [9].

Dalším aspektem je, že databáze spravující dokumentově orientované dokumenty nevyžadují normalizaci a to vzhledem k proměnlivému obsahu.

3.2.6.1 Textové nativní XML databáze

Textové nativní XML databáze ukládají XML jako text (BLOB, nebo CLOB). Pro takovéto databáze je běžná indexace, umožňující přesunout se kamkoliv do dokumentu. Pokud jsou data uložena na disku v jednom monolitu a v pořadí, ve kterém jsou indexována, je možné provést operace vyhledání a získání dotazovaného obsahu během

jednoho čtení. Tento model se stává méně efektivním při operacích, které nevrací data v původní podobě a vrací je v jiné formě [15]. Může dojít k situaci, kdy byl na disku monolit původních dat obklopen dalšími daty, změněná data se již nemohou vejít do původního oddílu a budou roztroušena na více místech.

3.2.6.2 Modelové nativní XML databáze

Modelové nativní XML databáze nepracují s XML dokumentem jako s textem, ale vytváří si podle XML dokumentu vlastní model a v něm dokument uloží (XDBM, Xindice). Existují dvě skupiny takovýchto databází, první vytváří svůj datový model postavený na modelu databáze, na které jsou postaveny, mají podobný výkon, jako ty, na nichž jsou postaveny a druhá skupina představuje databáze se svým vlastním datovým modelem, výkonem odpovídající textovým databázím [15].

3.2.7 Komprese XML

Jednou z hlavních nevýhod XML je neúspornost tohoto formátu pramenící z podstaty značkovacího jazyka. Jednou z cest, jak se vypořádat s tímto problémem je kompresní algoritmus uzpůsoben speciálně pro XML. Základní požadavky na kompresní algoritmus jsou bezztrátovost a dekomprese rychlejší než komprese. Běžné algoritmy pracují s XML jako s jednou skupinou dat, ačkoliv je XML tvořeno atributy/elementy, daty a jejich strukturou.

Moderní kompresní algoritmy pracující na bázi dynamických slovníků. V XML dokumentech dochází k častému opakování obsahu (názvy elementů a atributů, může dojít i k opakování hodnot), ale seznamy opakujících se slov se liší napříč XML dokumenty. Například elementy mohou být pojmenovány *firstname, FirstName, first_name, FIRSTNAME* ,atd., ale při použití předem daného seznamu nedokáže algoritmus rozlišit stejný význam různých textových řetězců.

Při výběru kompresního algoritmu je třeba brát na zřetel požadavky koncového uživatele. Pokud potřebujeme komprimovat online soubory, bude rozhodujícím faktorem rychlost dekomprese a nejvhodnějšími kandidáty budou XML-WRT a LZMA. Pokud je požadavkem dosažení nejlepšího kompresního poměru, upravené algoritmy typu LZ77 Deflate jsou nejvhodnějším řešením [3].

3.2.8 Grafické aplikace

Společnost Google vyvinula za spolupráce s Open Geospatial Konsorciem otevřený standard k zobrazování geografických dat, odvozený z XML, KML. Tento formát je využíván v aplikaci Google Earth, Google Maps a Google Maps for mobile. Jazyk samotný je formou XML s předem definovanými elementy [17].

4 Současný stav implementace XML

Programy běžně používané v podnikové praxi schopné vytvoření XML výstupu se dají rozdělit do dvou skupin, přičemž kritériem je elegance, s jakou je s XML pracováno. Mezi programy, které mají práci s XML dobře zvládnutou, patří: Microsoft Office 2007, SQL Server 2005, SQL Server 2008. Programový balík podporující XML, ale pokulhávající v praktickém použití je OpenOffice3.1.

Tato kapitola je zaměřena na praktické rady k využití XML v popisovaných situacích. Cílem zkoumání byly pouze možnosti využití XML ve jmenovaných programech, nikoliv však hodnocení samotných programů.

4.1 Souborové XML databáze

Pokud není k dispozici databázová aplikace, lze se uchýlit k využívání jednotlivých XML souborů jako databáze. Nejjednodušší formou XML databáze jsou samotné XML soubory, ať už jsou data uložena jako elementy, nebo jako atributy. V tomto modelu nelze vytvářet komplexní soustavy navzájem propojených tabulek, jak to umožňují databázové systémy, ale pouze jednotlivé a samostatné tabulky, které jsou členěné pomocí systému souborů.

Dotazování v takovéto databázi je omezeno na fulltextové vyhledávání v samotných souborech, případně vyhledávání v adresářích podle názvů, pod kterými byly soubory uloženy. Předpokládejme, že entitní integrita byla zajištěna při samotné tvorbě souborů v tabulkovém procesoru. Pokud je nutné zajistit entitní integritu, je na uživateli aby si zvolil příslušnou formu DTD, nebo schématu. Poté zbývá omezit jednotlivým uživatelům přístup k vybraným souborům. Toho lze docílit pomocí uživatelských účtů a jejich práv v příslušném operačním systému. Takovýto přístup je krajně nevhodný, protože jsou k dispozici open source databázové systémy, součásti kancelářských balíků a v rámci

předem daných mezí může uživatel využít „Enterprise Edition” SQL Serveru, nebo jiné relační databáze.

4.2 Microsoft Access 2007

Export obsahu databáze, ať už tabulek, nebo sestav je jednoduše proveden skrze použití pravého tlačítka myši, volbou položky exportovat a zvolení výstupu: XML soubor. Soubor lze exportovat jako surový XML – obsahuje pouze data a nikoliv údaje o barvě písma, barvě výplně, velikosti buňky, atd., nebo k němu lze připojit schéma, nebo XSL. Exportovaný soubor má strukturu, ve které jsou data uložena jako elementy a je datově orientovaný. Pokud bude exportována tabulka, která je již propojená s ostatními tabulkami, bude XML kód rozšířen o příslušné tagy.

Import dat lze provést volbou cílové tabulky, volby importovat a zvolením zdroje XML soubor, nebo v panelu externí data. Můžeme zvolit, zdali bude importována pouze struktura, struktura a data, nebo budou data připojena k existující tabulce - zde je třeba dbát na to, aby importovaný a existující soubor neobsahoval duplicitní hodnoty v indexu, primárním klíči, nebo relaci k čemuž může dojít, pokud budeme importovat tabulku exportovanou z jiné databáze. K importu dat z XML není vyžadována přítomnost XML schématu.

4.3 Microsoft Excel

Import dat je prováděn podobně, jako je tomu u Accesu z panelu Data a volbou externího zdroje. Avšak je třeba se mít na pozoru před tabulkami exportovanými z databáze, které obsahují NULL hodnoty. Konkrétní problémy jsou demonstrovány příkladu z obrázku 10, který reprezentuje část XML kódu, který má být importován do Excelu a obsahuje NULL hodnoty, v tomto případě vynechané elementy.

```
...
<objednavka>
                <element1>8</element1>
                <element3>161</element3>
                <element4>325</element4>
                <element5>1047</element5>
                <element6>2523</element6>
                <element7>1443</element7>
                <element8>3659</element8>
                <element9>5909</element9>
                <element10>8449</element10>
                <element11>16765</element11>
</objednavka>
...
<objednavka>
                <element1>7</element1>
                <element2>1</element2>
                <element4>62</element4>
                <element5>597</element5>
                <element6>2221</element6>
                <element7>2608</element7>
                <element8>5041</element8>
                <element9>7575</element9>
                <element10>6063</element10>
                <element11>20376</element11>
        </objednavka>
...
```

Obr. 10 Exportovaná tabulka obsahující NULL hodnoty

Zdroj: vlastní

Při importování souborů s NULL hodnotami dochází v Excelu k problémům projevujícím se chybovou zprávou, která nejasně udává svou příčinu, viz obrázek 11.

Obr. 11 Chybová zpráva

Zdroj: vlastní

Pro předejití všem problémům je vhodné nejprve v databázi zaměnit všechny NULL hodnoty na unikátní znaky, které se jinde v tabulce nevyskytují a po importu tyto znaky v Excelu nahradit přípustnými hodnotami. Excel nevyžaduje přítomnost DTD, nebo schématu k načtení XML dokumentu.

4.4 OpenOffice 3.1

Nenabízí srovnatelné využití XML jako Microsoft Office 2007. V databázové aplikaci nelze importovat ani exportovat jednotlivé tabulky prostřednictvím XML. Existuje pouze možnost uložit formulář, nebo sestavu jako Microsoft Word/Excel 2003 XML. Takovýto soubor není datově orientovaný ani strukturován, z datového pohledu obsahuje kód řadu zbytečností a jeho další zpracování je velmi obtížné, ne-li nemožné.

5 Porovnání zápisu dat formou elementů a atributů

Častou situací, ve které se uživatelé nachází, je použití XML k přenosu dat a uložení do programů odlišných od původce souboru. Volba spočívá v rozhodnutí se mezi uložením dat jako elementů a atributů. Oba přístupy jsou porovnány na následujících souborech představujících tabulky o 11 sloupcích a 100, 200, 300, 400, 500, 800, 1000, 1500, 2000, 3000, 5000 a 10000 řádcích obsahující náhodně generovaná čísla, viz tabulka 1. Vzájemně porovnávané soubory obsahují stejná data, liší se pouze formou zápisu. Cílem porovnání bylo zjistit, jak se jednotlivé přístupy k uchování XML dat vzájemně liší ve velikosti, kterou stejné soubory zaberou na disku.

Tab. 1 Obsah souborů

Sloupec	Interval náhodného čísla
1	1 - 10
2	1 - 20
3	20 - 200
4	50 - 500
5	100 - 1000
6	1000 - 2000
7	1000 - 3000
8	2000 - 4000
9	1000 - 8000
10	5000 - 10000
11	10000 - 20000

Zdroj: vlastní

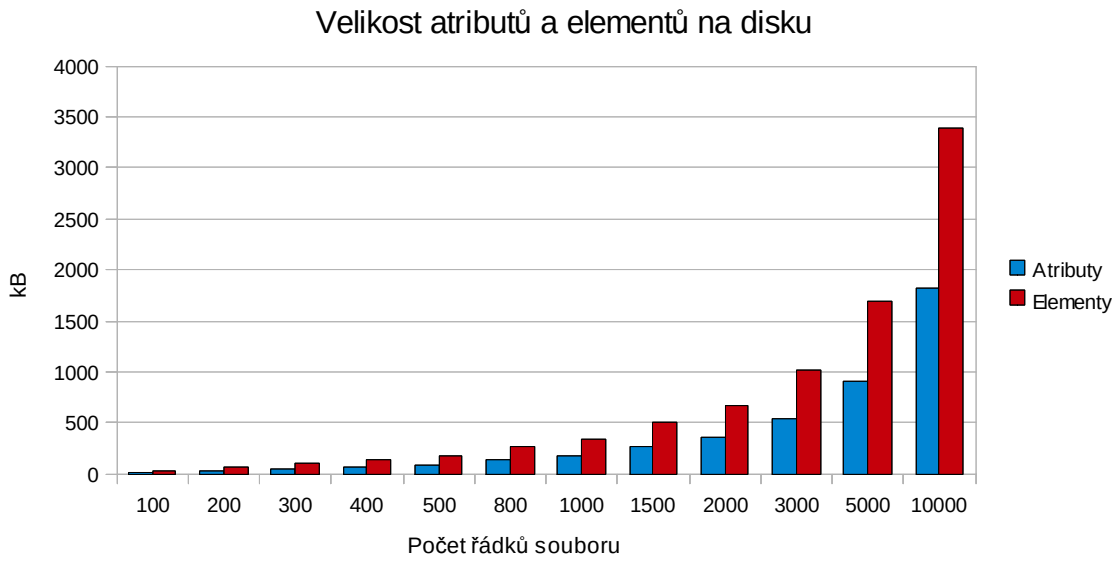
Velikosti jednotlivých souborů jsou zaznamenány v tabulce 2.

Tab. 2 Porovnání zápisu formou elementů a atributů

	Elementy		Atributy	
Počet řádků	Velikost [kB]	Velikost na disku [kB]	Velikost [kB]	Velikost na disku [kB]
100	33,9	34	18,2	20
200	67,7	68	36,5	38
300	101	102	54,7	56
400	135	136	73	74
500	169	170	91,2	92
800	270	272	145	146
1000	338	340	182	184
1500	508	510	273	274
2000	677	678	364	366
3000	1016	1016	547	548
5000	1694	1694	911	912
10000	3387	3388	1824	1824

Zdroj: vlastní

Operační systém Windows ukládá data do předem alokovaných oddílů, proto vzniká rozdíl mezi skutečnou velikostí dat a velikostí, kterou data zabírají na disku. Tento rozdíl je ovšem minimální a představuje zanedbatelné procento z kapacity pevných disků. V práci je dále používáno místo, které soubory zabírají na disku. Vzájemný poměr spotřebovaného místa na disku je znázorněn v grafu 1.



Graf 1 Velikost atributů a elementů na disku

Zdroj: vlastní

Zápis dat formou elementů má několik předností: je přehlednější než zápis formou atributů, snáze se edituje a oproti zápisu formou atributů je vhodný pro ukládání dokumentově orientovaných dokumentů, nebo dokumentů obsahující textová pole. Proti zápisu dat formou elementů hovoří fakt, že zabírá téměř dvakrát tolik místa, než zápis dat formou atributů.

5.1 Microsoft SQL Server 2008

Obsahem této podkapitoly je poskytnutí přehledu o nejefektivnějším způsobu nakládání s datově orientovanými dokumenty při jejich importu do SQL Serveru, přičemž kritérii hodnocení jsou doba operace, množství spotřebované paměti a využití procesoru. Váha jednotlivých kritérií nebyla předem pevně stanovena. Kritéria byla zvolena takto, protože dohromady vystihují nároky softwaru kladené na hardware. Původní záměr byl provést měření Microsoft SQL Serveru 2005 na denně používaném notebooku a velká část měření také byla provedena, ale došlo ke znemožnění dalších pokusů. Během provedených měření na SQL Serveru 2005 bylo dokončeno zkoumání metody 2 a tato metoda nedopadla při použití na SQL Serveru 2005 tak špatně, jako na SQL Serveru 2008. Postřehy z měření na

SQL Serveru 2005 jsou doplňovány k dílčím rozborům v kapitole 5.1.5 Porovnání jednotlivých metod.

SQL Server byl zvolen, protože byla možnost provést měření na tomto softwaru ve firmě Tarmac CZ a.s.

5.1.1 Metodika měření

Měření bylo prováděno na následujícím hardwaru:

- Procesor: Inter(R) Core(TM)2 CPU T7200 @ 2.00 GHz 2.00 GHz
- Paměť RAM: 1,00 GB
- Typ systému: 32 bitový operační systém
- Verze operačního systému: Windows Vista Business, Service Pack 2

Počítač byl zprovozněn pro účely měření, byla provedena nová instalace systému a jediný uživatelem instalovaný software byl SQL server 2008. K získání hodnot bylo použito systémového nástroje sledování spolehlivosti a výkonu, výsledky byly zaznamenávány a ukládány v automaticky vygenerovaných sestavách, které si lze prohlédnout na přiloženém nosiči v adresáři Měření/SQL2008.

Při měření byly spuštěny pouze programy nutné k provedení testu, a to:

- Microsoft SQL Server Management Studio Express
- Sledování spolehlivosti a výkonu
- Průzkumník Windows

Byly zvoleny tři vzájemně odlišné SQL příkazy k naplnění tabulek a jednotlivé soubory byly desetkrát importovány do SQL Serveru 2008 v SQL Server Management Studiu. Jednotlivé příkazy jsou pro potřeby práce pojmenovány metoda, ale nejedná se o normované postupy. Formy jednotlivých příkazů vzešly ze zkušeností nabytých v průběhu řízené praxe a konzultaci na fóru MSDN. V cílové databázi byla zřízena jediná, průběžně

rušená a znovu vytvářená tabulka, do které byla data směrována. Bylo sledováno, jak procesy *ssms.exe* a *sqlservr.exe* využívají procesor a jaké jsou jim přiděleny soukromé pracovní sady paměti. Procesům byly přiřazovány i sdílené pracovní sady paměti, tyto hodnoty lze najít v sestavách na nosiči, ale v práci nebyly uvažovány a to vzhledem k faktu, že nelze určit, jaké množství výkonu bylo z naměřeného čísla spotřebováno sledovaným procesem, nebo ostatními procesy které sdílely vyhrazenou kapacitu. Proces sqlservr.exe je procesem spouštěným SQL Serverem 2008. Proces ssms.exe je systémovým procesem, kterému se přiřazuje SQL Management Studio. Dále byla z SQL Server Management Studia odečítána a zaznamenávána doba potřebná k provedení požadovaného úkolu. Maximální velikost importovaného souboru byla zvolena podle velikosti denně zpracovávaných souborů ve společnosti Tarmac CZ a.s. Následující tři kapitoly obsahují výsledky měření, jejich zhodnocení následuje samostatně.

5.1.2 Metoda 1

Pro import dat do SQL serveru bylo využito příkazu na obrázku 12, data byla uložena jako atributy.

```
DECLARE @xml XML
SELECT @xml = x.y
FROM OPENROWSET( BULK 'e:\xml\100\atributy.xml', SINGLE_CLOB ) x(y)

INSERT INTO dbo.atributy100 ( element1, element2, element3, element4,
element5,   element6,   element7,   element8,   element9,   element10,
element11 )
SELECT
      x.y.value('@element1', 'INT'),
      x.y.value('@element2', 'INT'),
      x.y.value('@element3', 'INT'),
      x.y.value('@element4', 'INT'),
      x.y.value('@element5', 'INT'),
      x.y.value('@element6', 'INT'),
      x.y.value('@element7', 'INT'),
      x.y.value('@element8', 'INT'),
      x.y.value('@element9', 'INT'),
      x.y.value('@element10', 'INT'),
      x.y.value('@element11', 'INT')

FROM @xml.nodes('root/objednavka') x(y)
```

Obr. 12 Načtení XML obsahu metodou 1

Zdroj: http://social.msdn.microsoft.com/Forums/en/sqlxml/thread/584a6ab0-f4d7-49c2-b914-c9337ff7cd29

Bylo provedeno měření zatížení procesoru a to při importu souborů popsaných v tabulce 2. Výsledky jsou v tabulce 3. Naměřené hodnoty představují procentuální podíl na celkovém výkonu procesoru.

Tab. 3 Naměřené hodnoty při importu dat metodou 1

Velikost souboru [řádky]	Proces	Měření									
		1	2	3	4	5	6	7	8	9	10
100	ssms.exe	5,6	4,8	6,6	5,0	5,8	5,9	5,7	6,1	6,9	4,4
	sqlservr.exe	0,5	0,5	0,9	0,5	0,4	0,5	0,4	0,2	0,7	0,4
200	ssms.exe	5,9	5,3	5,8	5,7	5,3	4,8	7,0	5,4	5,3	5,5
	sqlservr.exe	0,8	0,4	0,9	0,7	1,0	0,8	1,0	0,9	0,5	0,8
300	ssms.exe	5,3	5,6	6,9	5,2	5,7	7,6	5,3	5,0	5,0	6,6
	sqlservr.exe	0,8	1,0	1,3	0,6	1,0	0,8	1,0	0,9	0,9	1,2
400	ssms.exe	6,3	6,2	6,0	5,6	5,5	7,3	5,5	5,0	5,7	5,4
	sqlservr.exe	1,0	1,1	0,8	1,1	1,5	0,9	1,2	1,1	1,1	1,0
500	ssms.exe	4,7	4,9	6,8	6,0	7,3	6,9	7,0	5,9	5,0	5,7
	sqlservr.exe	1,1	1,2	1,1	1,4	1,5	1,5	1,5	1,2	1,1	1,1
800	ssms.exe	5,4	6,8	4,5	5,3	5,3	4,5	6,3	5,9	3,5	6,6
	sqlservr.exe	1,8	2,1	1,7	2,0	1,8	1,6	1,7	2,2	1,1	2,0
1000	ssms.exe	5,9	5,5	7,2	5,5	7,5	5,9	4,6	6,4	6,8	6,1
	sqlservr.exe	2,5	1,9	2,0	2,3	2,8	2,1	1,9	2,4	2,1	2,0
1500	ssms.exe	4,8	6,2	7,1	7,5	4,5	6,1	4,2	5,7	6,2	6,1
	sqlservr.exe	2,2	2,8	3,0	3,1	2,5	3,3	2,5	3,4	3,2	3,2
2000	ssms.exe	6,1	5,1	6,2	6,2	5,6	5,5	5,7	6,7	7,2	5,9
	sqlservr.exe	4,0	3,5	4,2	4,4	3,7	3,8	3,8	4,7	4,2	4,0
3000	ssms.exe	5,3	4,6	5,4	6,6	5,4	6,2	6,2	5,9	4,5	4,5
	sqlservr.exe	4,8	4,6	5,2	5,4	4,3	5,2	6,1	5,5	5,3	5,2
5000	ssms.exe	4,8	5,1	5,5	5,0	6,6	5,3	5,5	5,5	5,1	6,7
	sqlservr.exe	7,4	7,2	7,8	8,4	10,6	8,9	9,4	8,5	9,2	8,9
10000	ssms.exe	4,0	4,3	4,4	4,1	4,0	3,8	4,2	4,6	4,7	5,1
	sqlservr.exe	12,9	13,8	13,5	14,2	12,4	12,9	14,7	13,9	13,1	14,1

Zdroj: vlastní

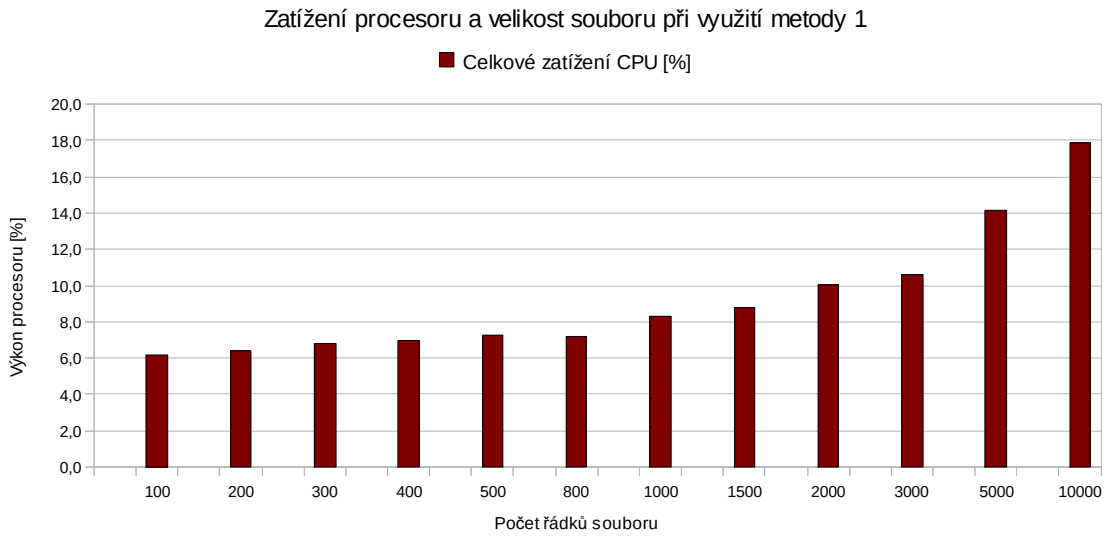
Hodnoty byly dále zpracovány, byl vypočten aritmetický průměr pro jednotlivé procesy a součtem hodnot obou procesů bylo určeno celkové zatížení procesoru. Pro jednotlivé procesy byla vypočtena směrodatná odchylka, viz tabulka 4.

Tab. 4 Hodnoty odvozené z tabulky 3

Velikost souboru [řádky]	Proces	Průměrné zatížení CPU [%]	Směrodatná odchylka	Směrodatná odchylka v %	Celkové zatížení CPU [%]
100	ssms.exe	5,68	0,78	13,71	6,18
	sqlservr.exe	0,50	0,19	37,71	
200	ssms.exe	5,60	0,58	10,41	6,38
	sqlservr.exe	0,78	0,20	25,50	
300	ssms.exe	5,82	0,90	15,45	6,77
	sqlservr.exe	0,95	0,20	21,20	
400	ssms.exe	5,85	0,64	10,97	6,93
	sqlservr.exe	1,08	0,19	17,35	
500	ssms.exe	6,02	0,95	15,79	7,29
	sqlservr.exe	1,27	0,18	14,40	
800	ssms.exe	5,41	1,04	19,20	7,21
	sqlservr.exe	1,80	0,31	17,37	
1000	ssms.exe	6,14	0,87	14,14	8,34
	sqlservr.exe	2,20	0,29	13,38	
1500	ssms.exe	5,84	1,07	18,34	8,76
	sqlservr.exe	2,92	0,40	13,77	
2000	ssms.exe	6,02	0,61	10,11	10,05
	sqlservr.exe	4,03	0,36	8,84	
3000	ssms.exe	5,46	0,76	13,93	10,62
	sqlservr.exe	5,16	0,50	9,72	
5000	ssms.exe	5,51	0,65	11,71	14,14
	sqlservr.exe	8,63	1,01	11,75	
10000	ssms.exe	4,32	0,39	9,05	17,87
	sqlservr.exe	13,55	0,72	5,28	

Zdroj: vlastní

Jednotlivá měření vykazují proměnlivou míru variability, která klesá s narůstající velikostí souboru. Vzájemné odchylky měření pro jednotlivé skupiny jsou přijatelné pro soubory větší než 1000 řádků. Graf 2 představuje průběh sledované veličiny pro jednotlivé velikosti souborů.



Graf 2 Závislost využití procesoru a velikosti souboru při importu dat metodou 1

Zdroj: vlastní

Doba jednotlivých operací je v tabulce 5, jednotlivá pozorování jsou v sekundách.

Tab. 5 Naměřené hodnoty při importu dat metodou 1

Velikost souboru [řádky]	Měření									
	1	2	3	4	5	6	7	8	9	10
100	0,116	0,101	0,135	0,103	0,098	0,100	0,104	0,184	0,110	0,101
200	0,139	0,134	0,146	0,149	0,141	0,143	0,145	0,140	0,141	0,148
300	0,177	0,188	0,190	0,180	0,177	0,264	0,182	0,165	0,181	0,173
400	0,220	0,212	0,212	0,207	0,215	0,302	0,202	0,208	0,203	0,214
500	0,254	0,238	0,316	0,238	0,237	0,249	0,244	0,253	0,240	0,241
800	0,349	0,328	0,336	0,344	0,399	0,330	0,349	0,350	0,445	0,405
1000	0,394	0,390	0,393	0,395	0,385	0,401	0,480	0,384	0,434	0,382
1500	0,645	0,635	0,542	0,642	0,538	0,531	0,533	0,634	0,526	0,569
2000	0,681	0,687	0,687	0,697	0,683	0,693	0,731	0,700	0,692	0,687
3000	1,309	0,985	0,971	0,984	0,997	0,990	0,967	0,995	0,992	0,979
5000	1,568	1,592	1,547	1,571	1,568	1,584	1,570	1,589	1,582	1,567
10000	3,168	3,069	3,078	3,099	3,013	3,022	3,431	3,048	3,082	3,136

Zdroj: vlastní

Z naměřených hodnot v tabulce 5 byl pro jednotlivé velikosti vypočten aritmetický průměr a směrodatná odchylka. Vypočtené hodnoty jsou v tabulce 6.

Tab. 6 Hodnoty odvozené z tabulky 5

Velikost souboru [řádky]	Aritmetický průměr měření	Směrodatná odchylka	Směrodatná odchylka v %
100	0,115	0,027	23,071
200	0,143	0,005	3,191
300	0,188	0,028	14,777
400	0,220	0,030	13,443
500	0,251	0,024	9,434
800	0,364	0,039	10,752
1000	0,404	0,031	7,578
1500	0,580	0,053	9,070
2000	0,694	0,014	2,070
3000	1,017	0,103	10,139
5000	1,574	0,013	0,844
10000	3,115	0,121	3,881

Zdroj: vlastní

Variabilita měření je proměnlivá a klesá s rostoucí velikostí souboru.

Dále bylo sledováno, jak procesy *ssms.exe* a *sqlservr.exe* využívají paměť, konkrétně jak velké jsou jejich soukromé pracovní sady. Naměřené hodnoty jsou v tabulce 7. Jednotlivé hodnoty jsou v KB.

Tab. 7 Využití paměti, metoda 1

Velikost souboru [řádky]	Proces	Měření									
		1	2	3	4	5	6	7	8	9	10
100	ssms.exe	35856	35832	35848	37700	36320	36336	36224	37296	37300	36968
	sqlservr.exe	66788	66788	66776	66784	66784	66784	66796	66784	66796	66796
200	ssms.exe	40260	40284	40436	40416	40404	36580	35868	35860	35856	36228
	sqlservr.exe	66808	66796	66904	66796	66796	66796	66796	66904	66808	66796
300	ssms.exe	39240	39580	37952	35812	35916	35892	37664	36516	36212	36236
	sqlservr.exe	66804	66820	66812	66812	66812	66824	66824	66812	66824	66824
400	ssms.exe	39540	37732	37012	37428	37428	37424	40692	38524	38700	38352
	sqlservr.exe	66824	66824	66824	66836	66932	66824	66768	66768	66768	66780
500	ssms.exe	46688	41424	41424	41308	41332	41308	38240	36544	36032	36020
	sqlservr.exe	66768	66768	66768	66780	66780	66768	66828	66800	66812	66812
800	ssms.exe	43968	39196	38840	38848	38916	37276	35860	35860	35860	35896
	sqlservr.exe	66788	66792	66940	66804	66804	66940	66796	66808	66808	66808
1000	ssms.exe	41668	37980	37984	37972	37968	37964	38112	39952	38532	38320
	sqlservr.exe	66800	66812	66812	66800	66800	66800	66800	66800	66800	66800
1500	ssms.exe	46652	39944	39944	39956	39936	39948	40012	40068	40052	39972
	sqlservr.exe	66800	66800	66808	66808	66808	66800	66944	66800	66800	66800
2000	ssms.exe	38948	38960	38936	38940	38948	38932	39012	39172	39440	39748
	sqlservr.exe	66808	38960	66812	66812	66812	66812	66812	66812	66812	66812
3000	ssms.exe	45976	39040	39040	38832	36432	36440	36444	36420	36556	36612
	sqlservr.exe	66964	66812	66812	66824	66812	66812	66828	66828	66812	66812
5000	ssms.exe	39868	39848	39664	39684	39648	39656	39776	39864	39960	40344
	sqlservr.exe	66816	66816	66828	66828	66816	66968	66816	66816	66976	66816
10000	ssms.exe	44528	39140	39124	38920	38928	38916	38920	37720	36436	36440
	sqlservr.exe	66892	66820	66972	66992	66972	66840	67000	67004	67376	67876

Zdroj: vlastní

Hodnoty z tabulky 7 byly dále zpracovány, byl vypočten aritmetický průměr pro jednotlivé procesy a součtem hodnot obou procesů bylo určeno celkové využití paměti. Pro jednotlivé procesy byla vypočtena směrodatná odchylka, viz tabulka 8.

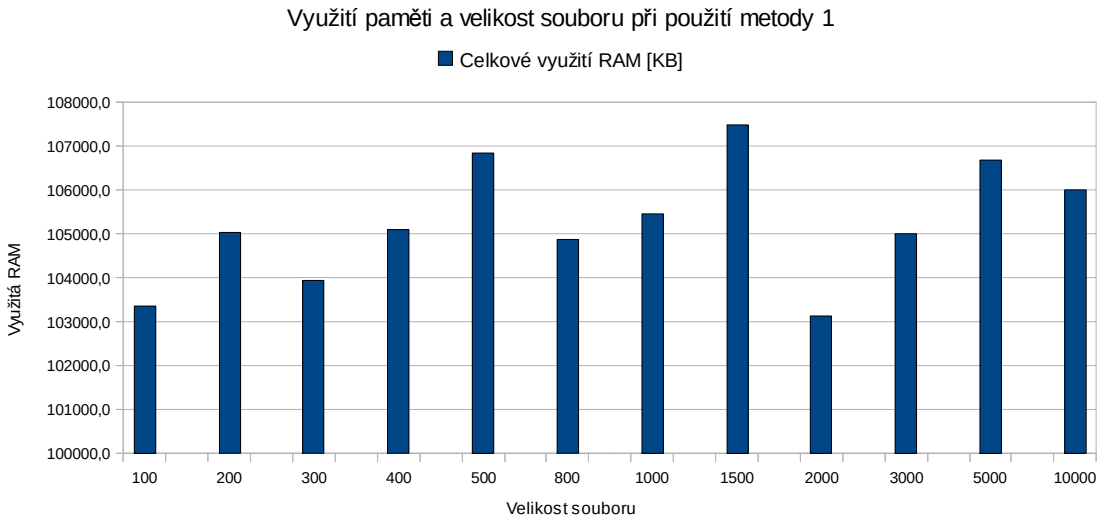
Tab. 8 Hodnoty odvozené z tabulky 7

Velikost souboru [řádky]	Proces	Průměrné využití RAM [KB]	Směrodatná odchylka	Směrodatná odchylka v %	Celkové využití RAM [KB]
100	ssms.exe	36568,00	691,85	1,89	103355,60
	sqlservr.exe	66787,60	6,65	0,01	
200	ssms.exe	38219,20	2267,46	5,93	105039,20
	sqlservr.exe	66820,00	44,54	0,07	
300	ssms.exe	37102,00	1420,88	3,83	103918,80
	sqlservr.exe	66816,80	7,25	0,01	
400	ssms.exe	38283,20	1142,08	2,98	105098,00
	sqlservr.exe	66814,80	49,78	0,07	
500	ssms.exe	40032,00	3343,28	8,35	106820,40
	sqlservr.exe	66788,40	22,66	0,03	
800	ssms.exe	38052,00	2538,02	6,67	104880,80
	sqlservr.exe	66828,80	59,02	0,09	
1000	ssms.exe	38645,20	1224,90	3,17	105447,60
	sqlservr.exe	66802,40	5,06	0,01	
1500	ssms.exe	40648,40	2109,98	5,19	107465,20
	sqlservr.exe	66816,80	44,85	0,07	
2000	ssms.exe	39103,60	277,85	0,71	103130,00
	sqlservr.exe	64026,40	8807,44	13,76	
3000	ssms.exe	38179,20	2980,91	7,81	105010,80
	sqlservr.exe	66831,60	47,04	0,07	
5000	ssms.exe	39831,20	210,61	0,53	106680,80
	sqlservr.exe	66849,60	64,72	0,10	
10000	ssms.exe	38907,20	2243,48	5,77	105981,60
	sqlservr.exe	67074,40	320,54	0,48	

Zdroj: vlastní

Jednotlivá pozorování vykazují velmi nízkou míru variability. Jedinou výjimkou je pozorování procesu *sqlservr.exe* pro soubor o velikosti 2000 řádků.

Graf 3 představuje průběh sledované veličiny pro jednotlivé velikosti souborů.



Graf 3 Závislost využití paměti RAM a velikosti souboru při využití metody 1

Zdroj: vlastní

SQL Server spotřeboval stálé množství paměti pro operaci se sledovanými soubory.

5.1.3 Metoda 2

Pro import dat do SQL serveru bylo využito příkazu na obrázku 13, data byla uložena jako elementy.

```
DECLARE @xml XML
SELECT @xml = x.y
FROM OPENROWSET( BULK 'e:\XML\1000\elementy.xml', SINGLE_CLOB ) x(y)


INSERT INTO dbo.elementy100 ( element1, element2, element3,
element4, element5, element6, element7, element8, element9,
element10, element11 )
SELECT
      x.y.value('element1[1]', 'INT'),
      x.y.value('element2[1]', 'INT'),
      x.y.value('element3[1]', 'INT'),
      x.y.value('element4[1]', 'INT'),
      x.y.value('element5[1]', 'INT'),
      x.y.value('element6[1]', 'INT'),
      x.y.value('element7[1]', 'INT'),
      x.y.value('element8[1]', 'INT'),
      x.y.value('element9[1]', 'INT'),
      x.y.value('element10[1]', 'INT'),
      x.y.value('element11[1]', 'INT')
FROM @xml.nodes('root/objednavka') x(y)
```

Obr. 13 Načtení XML obsahu metodou 2

Zdroj: http://social.msdn.microsoft.com/Forums/en/sqlxml/thread/584a6ab0-f4d7-49c2-b914-c9337ff7cd29

Bylo provedeno měření zatížení procesoru a to při importu souborů popsaných v tabulce 2. Výsledky jsou v tabulce 9. Naměřené hodnoty představují procentuální podíl na celkovém výkonu procesoru.

Tab. 9 Naměřené hodnoty při importu dat metodou 2

Velikost souboru [řádky]	Proces	Měření									
		1	2	3	4	5	6	7	8	9	10
100	ssms.exe	3,5	2,4	3,3	3,3	4,0	3,7	3,5	3,5	3,3	4,1
	sqlservr.exe	13,1	12,5	13,2	14,0	15,0	14,0	13,0	14,8	14,0	15,5
200	ssms.exe	1,6	1,7	1,7	1,8	1,8	2,0	1,7	1,7	1,8	1,4
	sqlservr.exe	26,7	27,0	28,8	28,7	26,4	28,4	29,7	28,3	28,3	27,2
300	ssms.exe	0,7	0,9	1,0	1,0	1,1	1,1	0,9	0,8	0,8	0,9
	sqlservr.exe	28,4	28,8	34,3	34,3	35,1	37,2	34,6	32,5	31,6	32,3
400	ssms.exe	0,6	0,9	0,8	0,7	0,6	0,5	0,8	0,6	0,7	0,7
	sqlservr.exe	38,9	41,2	37,1	40,6	34,5	41,3	39,1	40,6	40,0	39,0
500	ssms.exe	0,5	0,4	0,4	0,5	0,3	0,4	0,3	0,5	0,4	0,4
	sqlservr.exe	40,5	43,2	43,0	42,6	40,4	36,1	40,2	41,5	42,1	41,9
800	ssms.exe	0,2	0,3	0,2	0,2	0,2	0,2	0,3	0,2	0,2	0,2
	sqlservr.exe	45,3	40,4	44,6	45,1	44,7	45,2	44,5	44,8	44,2	45,3
1000	ssms.exe	0,2	0,2	0,1	0,1	0,2	0,1	0,1	0,2	0,1	0,2
	sqlservr.exe	44,5	45,4	44,8	45,5	45,0	45,6	46,1	44,7	45,2	44,7

Zdroj: vlastní

Bylo provedeno měření pouze pro soubory do 1000 řádků a to vzhledem k nárokům na výpočetní výkon a celkovou dobu operace, viz tabulka 11.

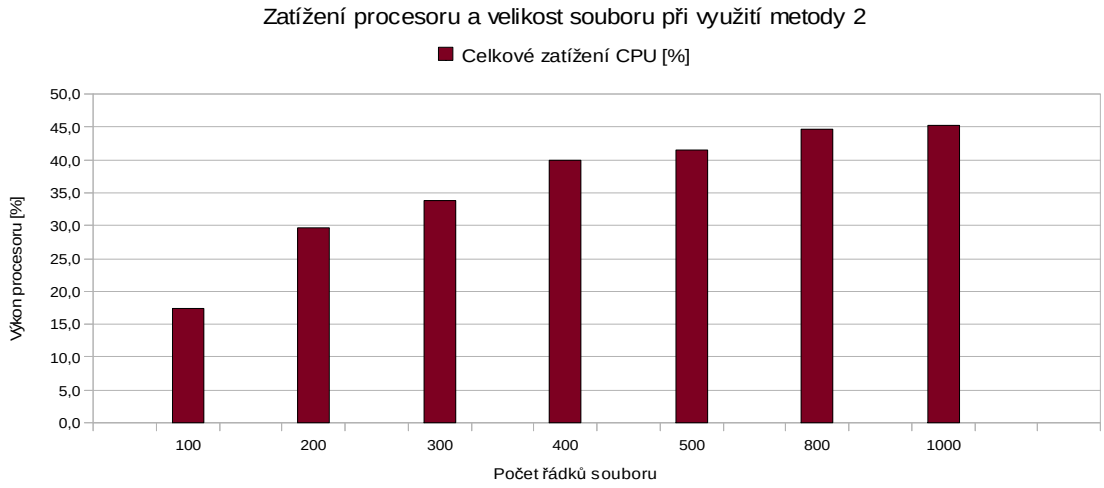
Hodnoty byly dále zpracovány, byl vypočten aritmetický průměr pro jednotlivé procesy a součtem hodnot obou procesů bylo určeno celkové zatížení procesoru. Pro jednotlivé procesy byla vypočtena směrodatná odchylka, viz tabulka 10.

Tab. 10 Hodnoty odvozené z tabulky 9

Velikost souboru [řádky]	Proces	Průměrné zatížení CPU [%]	Směrodatná odchylka	Směrodatná odchylka v %	Celkové zatížení CPU [%]
100	ssms.exe	3,46	0,47	13,50	17,37
	sqlservr.exe	13,91	0,97	6,99	
200	ssms.exe	1,72	0,15	9,01	29,67
	sqlservr.exe	27,95	1,07	3,81	
300	ssms.exe	0,92	0,13	14,31	33,83
	sqlservr.exe	32,91	2,78	8,44	
400	ssms.exe	0,69	0,12	17,35	39,92
	sqlservr.exe	39,23	2,10	5,35	
500	ssms.exe	0,41	0,07	18,00	41,56
	sqlservr.exe	41,15	2,07	5,04	
800	ssms.exe	0,22	0,04	19,17	44,63
	sqlservr.exe	44,41	1,46	3,28	
1000	ssms.exe	0,15	0,05	35,14	45,30
	sqlservr.exe	45,15	0,50	1,11	

Zdroj: vlastní

Pozorování procesu *ssms.exe* vykazují vyšší míru variability, ale podíl procesu na výkonu procesoru nedosahuje podstatných hodnot. Proces *sqlservr.exe* spotřeboval značné množství výpočetního výkonu a variabilita jednotlivých pozorování je nízká. Graf 4 představuje průběh sledované veličiny pro jednotlivé velikosti souborů.



Graf 4 Závislost využití procesoru a velikosti souboru při importu dat metodou 2

Zdroj: vlastní

Doba jednotlivých operací je v tabulce 11, jednotlivá pozorování jsou v sekundách.

Tab. 11 Naměřené hodnoty při importu dat metodou 2

Velikost souboru [řádky]	Měření									
	1	2	3	4	5	6	7	8	9	10
100	2,97	2,76	2,73	2,76	2,72	2,74	2,84	2,74	2,76	2,76
200	9,68	9,71	9,73	9,80	9,68	9,67	9,71	9,66	9,63	9,63
300	21,86	20,69	21,10	20,64	21,34	21,21	20,94	20,66	20,83	21,11
400	36,33	36,33	36,45	36,21	36,34	35,96	36,69	36,46	36,50	36,38
500	56,67	56,08	56,05	55,79	56,77	56,31	56,46	55,94	56,36	56,43
800	141,44	141,73	142,43	142,23	140,67	141,76	140,70	142,24	142,89	140,25
1000	222,37	219,75	218,73	219,86	221,47	220,96	220,36	218,80	219,26	219,15

Zdroj: vlastní

Z naměřených hodnot byl pro jednotlivé velikosti vypočten aritmetický průměr a směrodatná odchylka. Vypočtené hodnoty jsou v tabulce 12.

Tab. 12 Hodnoty odvozené z tabulky 11

Velikost souboru [řádky]	Aritmetický průměr měření	Směrodatná odchylka	Směrodatná odchylka v %
100	2,78	0,08	2,77
200	9,69	0,05	0,52
300	21,04	0,38	1,79
400	36,36	0,19	0,53
500	56,28	0,32	0,56
800	141,63	0,86	0,61
1000	220,07	1,21	0,55

Zdroj: vlastní

Hodnoty v tabulce 12 vykazují nízkou míru variability.

Dále bylo sledováno, jak procesy *ssms.exe* a *sqlservr.exe* využívají paměť, konkrétně jak velké jsou jejich soukromé pracovní sady. Naměřené hodnoty jsou v tabulce 13. Jednotlivé hodnoty jsou v KB.

Tab. 13 Vytížení paměti, metoda 2

Velikost souboru [řádky]	Proces	Měření									
		1	2	3	4	5	6	7	8	9	10
100	ssms.exe	36260	34024	34008	33828	35848	34712	34696	34536	34532	34556
	sqlservr.exe	57280	57736	58180	58648	59144	59480	59968	60308	60452	60424
200	ssms.exe	35652	34076	34044	34084	34200	34388	34532	36336	35608	35592
	sqlservr.exe	61708	62204	62740	63360	63508	63412	63432	63600	63620	63620
300	ssms.exe	35944	34144	34216	34408	34632	34820	35068	35196	35420	35688
	sqlservr.exe	63680	63680	63824	63824	63700	63704	63876	63704	63704	63848
400	ssms.exe	36576	34196	34472	34764	35040	35168	36812	34904	34888	34920
	sqlservr.exe	63908	63908	64052	63920	63944	63968	64008	64008	64008	64016
500	ssms.exe	36284	34452	35040	35016	34264	34312	34352	34364	34340	34388
	sqlservr.exe	64064	64064	64064	64268	64152	64176	64212	64236	64232	64352
800	ssms.exe	40744	35620	42200	42196	42228	42256	42320	42640	42384	42448
	sqlservr.exe	64472	66048	66252	66396	66340	66400	66432	66580	66512	66568
1000	ssms.exe	43480	44924	46304	46936	38680	38292	38328	38400	38440	38444
	sqlservr.exe	68024	69320	69552	69616	69844	69764	69812	69876	70020	70000

Zdroj: vlastní

Hodnoty byly dále zpracovány, byl vypočten aritmetický průměr pro jednotlivé procesy a součtem hodnot obou procesů bylo určeno celkové využití paměti. Pro jednotlivé procesy byla vypočtena směrodatná odchylka, viz tabulka 14.

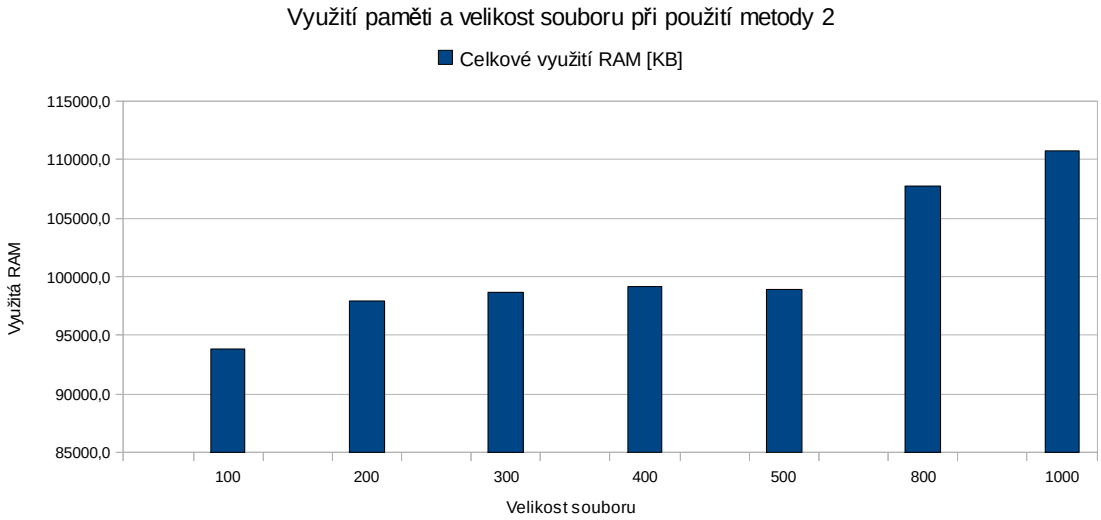
Tab. 14 Hodnoty odvozené z tabulky 13

Velikost souboru [řádky]	Proces	Průměrné využití RAM [KB]	Směrodatná odchylka	Směrodatná odchylka v %	Celkové využití RAM [KB]
100	ssms.exe	34700,00	783,42	2,26	93862,00
	sqlservr.exe	59162,00	1160,70	1,96	
200	ssms.exe	34851,20	853,04	2,45	97971,60
	sqlservr.exe	63120,40	674,81	1,07	
300	ssms.exe	34953,60	617,17	1,77	98708,00
	sqlservr.exe	63754,40	78,09	0,12	
400	ssms.exe	35174,00	850,05	2,42	99148,00
	sqlservr.exe	63974,00	51,51	0,08	
500	ssms.exe	34681,20	631,05	1,82	98863,20
	sqlservr.exe	64182,00	97,37	0,15	
800	ssms.exe	41503,60	2131,09	5,13	107703,60
	sqlservr.exe	66200,00	627,23	0,95	
1000	ssms.exe	41222,80	3713,67	9,01	110805,60
	sqlservr.exe	69582,80	587,44	0,84	

Zdroj: vlastní

Jednotlivá pozorování vykazují velmi nízkou míru variability.

Graf 5 představuje průběh sledované veličiny pro jednotlivé velikosti souborů.



Graf 5 Závislost využití paměti RAM a velikosti souboru při využití metody 2

Zdroj: vlastní

Množství vyžadované paměti narůstá s velikostí souborů.

5.1.4 Metoda 3

Vzhledem k nevyhovujícím dobám operace s metodou 2 byl zvolen jiný SQL příkaz. Pro import dat do SQL serveru bylo využito příkazu na obrázku 14, data byla uložena jako elementy.

```
declare @X xml
select @X = X.C
      from openrowset(bulk
      'e:\xml\100\elementy.xml',
single_blob) AS X(C)

insert into petr.dbo.elementy100
      select
C.value('(./element1/text())[1]','int') as 'element1'
,c.value('(./element2/text())[1]','int') as 'element2'
,c.value('(./element3/text())[1]','int') as 'element3'
,c.value('(./element4/text())[1]','int') as 'element4'
,c.value('(./element5/text())[1]', 'int') as 'element5'
,c.value('(./element6/text())[1]', 'int') as 'element6'
,c.value('(./element7/text())[1]', 'int') as 'element7'
,c.value('(./element8/text())[1]', 'int') as 'element8'
,c.value('(./element9/text())[1]', 'int') as 'element9'
,c.value('(./element10/text())[1]', 'int') as 'element10'
,c.value('(./element11/text())[1]', 'int') as 'element11'
from @x.nodes('/root/objednavka') T(C)

select
C.value ('*[1]', 'int') as 'element1'
,c.value ('*[2]', 'int') as 'element2'
,c.value ('*[3]', 'int') as 'element3'
,c.value ('*[4]', 'int') as 'element4'
,c.value ('*[5]', 'int') as 'element5'
,c.value ('*[6]', 'int') as 'element6'
,c.value ('*[7]', 'int') as 'element7'
,c.value ('*[8]', 'int') as 'element8'
,c.value ('*[9]', 'int') as 'element9'
,c.value ('*[10]', 'int') as 'element10'
,c.value ('*[11]', 'int') as 'element11'

from @x.nodes('/root/objednavka') T(C)
```

Obr. 14 Načtení XML obsahu metodou 3

Zdroj: http://social.msdn.microsoft.com/Forums/en/sqlxml/thread/584a6ab0-f4d7-49c2-b914-c9337ff7cd29

Bylo provedeno měření zatížení procesoru a to při importu souborů popsaných v tabulce 2. Výsledky jsou v tabulce 15. Naměřené hodnoty představují procentuální podíl na celkovém výkonu procesoru.

Tab. 15 Naměřené hodnoty při importu dat metodou 3

Velikost souboru [řádky]	Proces	Měření									
		1	2	3	4	5	6	7	8	9	10
100	ssms.exe	5,6	6,2	7,1	6,0	7,1	6,0	9,7	8,0	8,1	6,6
	sqlservr.exe	1,4	2,3	1,6	1,8	2,0	1,8	2,8	2,6	2,2	1,9
200	ssms.exe	7,1	7,2	8,0	7,6	7,4	7,6	5,8	7,6	6,5	6,5
	sqlservr.exe	3,0	3,1	3,1	3,5	3,0	2,9	2,9	3,2	2,7	2,6
300	ssms.exe	9,4	5,4	6,7	5,9	9,4	7,9	7,2	6,7	7,1	5,9
	sqlservr.exe	4,5	2,6	3,7	3,1	4,4	4,0	3,4	4,1	3,8	3,3
400	ssms.exe	7,6	5,9	7,1	6,8	5,6	7,9	6,4	6,6	6,2	6,3
	sqlservr.exe	4,8	3,8	4,4	4,6	4,5	4,7	4,5	4,6	4,2	4,0
500	ssms.exe	6,8	6,1	7,3	8,1	7,2	6,4	8,4	7,7	6,5	6,1
	sqlservr.exe	5,2	4,9	5,7	6,2	5,3	4,8	6,9	5,6	5,2	5,3
800	ssms.exe	7,3	8,7	6,8	7,1	7,3	7,2	7,1	6,2	7,5	8,4
	sqlservr.exe	6,9	8,3	7,1	8,6	7,6	7,9	9,9	7,6	6,7	7,9
1000	ssms.exe	5,7	7,0	6,1	6,4	6,8	6,8	6,9	5,6	5,5	7,4
	sqlservr.exe	7,3	8,5	8,4	8,0	8,7	9,3	8,7	7,7	9,5	9,6
1500	ssms.exe	6,2	5,7	4,2	6,6	6,6	6,7	6,3	6,0	6,1	6,2
	sqlservr.exe	11,8	10,2	7,4	12,5	12,8	12,9	12,9	11,0	12,3	12,2
2000	ssms.exe	6,3	4,9	4,8	5,5	6,6	5,9	5,3	6,0	5,9	5,8
	sqlservr.exe	14,9	11,4	12,1	16,8	16,1	13,1	12,5	15,0	13,3	13,5
3000	ssms.exe	6,4	4,5	4,8	6,2	5,3	5,5	4,9	5,7	5,9	6,1
	sqlservr.exe	20,0	16,1	19,0	18,2	17,6	18,9	17,3	20,6	20,6	19,7
5000	ssms.exe	4,9	4,2	4,1	4,4	4,2	4,1	4,9	4,8	4,2	4,4
	sqlservr.exe	24,8	23,1	24,0	24,7	24,5	22,8	23,3	25,4	23,9	25,2
10000	ssms.exe	3,1	3,8	3,0	2,7	2,9	2,8	3,4	2,7	3,0	3,1
	sqlservr.exe	28,7	29,9	30,7	28,1	28,7	30,8	29,2	31,5	30,8	31,0

Zdroj: vlastní

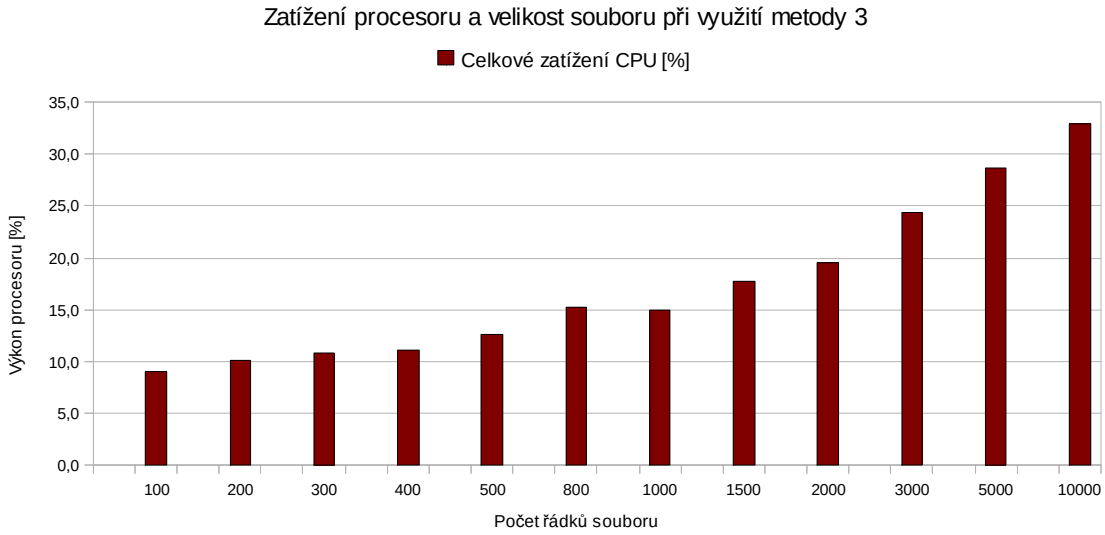
Hodnoty byly dále zpracovány, byl vypočten aritmetický průměr pro jednotlivé procesy a součtem hodnot obou procesů bylo určeno celkové zatížení procesoru. Pro jednotlivé procesy byla vypočtena směrodatná odchylka, viz tabulka 16.

Tab. 16 Hodnoty odvozené z tabulky 15

Velikost souboru [řádky]	Proces	Průměrné zatížení CPU [%]	Směrodatná odchylka	Směrodatná odchylka v %	Celkové zatížení CPU [%]
100	ssms.exe	7,04	1,26	17,88	9,08
	sqlservr.exe	2,04	0,44	21,45	
200	ssms.exe	7,13	0,67	9,42	10,13
	sqlservr.exe	3,00	0,25	8,46	
300	ssms.exe	7,16	1,39	19,35	10,85
	sqlservr.exe	3,69	0,60	16,18	
400	ssms.exe	6,64	0,73	10,93	11,05
	sqlservr.exe	4,41	0,32	7,21	
500	ssms.exe	7,06	0,82	11,59	12,57
	sqlservr.exe	5,51	0,63	11,49	
800	ssms.exe	7,36	0,72	9,84	15,21
	sqlservr.exe	7,85	0,94	11,92	
1000	ssms.exe	6,42	0,66	10,33	14,99
	sqlservr.exe	8,57	0,76	8,89	
1500	ssms.exe	6,06	0,72	11,90	17,66
	sqlservr.exe	11,60	1,72	14,79	
2000	ssms.exe	5,70	0,58	10,13	19,57
	sqlservr.exe	13,87	1,77	12,73	
3000	ssms.exe	5,53	0,64	11,66	24,33
	sqlservr.exe	18,80	1,50	7,95	
5000	ssms.exe	4,42	0,33	7,37	28,59
	sqlservr.exe	24,17	0,90	3,71	
10000	ssms.exe	3,05	0,34	11,06	32,99
	sqlservr.exe	29,94	1,18	3,96	

Zdroj: vlastní

Operace s malými soubory vykazují vyšší variabilitu, než operace se soubory s více řádky, zejména pozorování pro soubory o 100 a 300 řádcích. Graf 6 představuje průběh sledované veličiny pro jednotlivé velikosti souborů.



Graf 6 Závislost využití procesoru a velikosti souboru při využití metody 3

Zdroj: vlastní

Doba jednotlivých operací je v tabulce 17, jednotlivá pozorování jsou v sekundách.

Tab. 17 Naměřené hodnoty při importu dat metodou 3

Velikost souboru [řádky]	Měření									
	1	2	3	4	5	6	7	8	9	10
100	0,767	0,676	0,649	0,673	0,685	0,777	0,762	0,744	0,713	0,725
200	0,701	0,735	0,715	0,818	0,708	0,730	0,696	0,738	0,699	0,705
300	0,864	0,776	0,873	0,783	0,801	0,772	0,839	0,871	0,987	0,799
400	0,964	0,929	1,010	0,932	1,074	0,940	0,926	0,927	0,940	0,951
500	1,118	1,124	1,066	1,079	1,070	1,069	1,076	1,093	1,148	1,134
800	1,530	1,561	1,511	1,493	1,544	1,529	1,548	1,548	1,506	1,523
1000	1,797	1,806	1,830	1,788	1,855	1,820	1,851	1,806	1,843	1,836
1500	2,536	2,534	2,578	2,560	2,546	2,864	2,556	2,544	2,542	2,588
2000	3,589	3,247	3,267	3,357	3,292	3,265	3,341	3,281	3,197	3,296
3000	4,710	5,040	4,964	4,816	4,716	4,793	4,789	4,668	4,885	4,788
5000	7,621	7,721	7,597	7,910	7,866	8,026	7,828	7,722	7,690	7,677
10000	15,398	14,927	15,482	15,092	14,971	15,081	15,490	15,621	15,426	15,288

Zdroj: vlastní

Z naměřených hodnot byl pro jednotlivé velikosti vypočten aritmetický průměr a směrodatná odchylka. Vypočtené hodnoty jsou v tabulce 18.

Tab. 18 Hodnoty odvozené z tabulky 17

Velikost souboru [řádky]	Aritmetický průměr	Směrodatná odchylka	Směrodatná odchylka v %
100	0,717	0,045	6,271
200	0,725	0,036	5,003
300	0,837	0,066	7,898
400	0,959	0,048	4,960
500	1,098	0,031	2,781
800	1,529	0,021	1,405
1000	1,823	0,023	1,282
1500	2,585	0,100	3,856
2000	3,313	0,107	3,227
3000	4,817	0,116	2,417
5000	7,766	0,137	1,767
10000	15,278	0,243	1,591

Zdroj: vlastní

Hodnoty v tabulce 18 vykazují nízkou míru variability.

Dále bylo sledováno, jak procesy *ssms.exe* a *sqlservr.exe* využívají paměť, konkrétně jak velké jsou jejich soukromé pracovní sady. Naměřené hodnoty jsou v tabulce 19. Jednotlivé hodnoty jsou v KB.

Tab. 19 Využití paměti, metoda 3

Velikost souboru [řádky]	Proces	Měření									
		1	2	3	4	5	6	7	8	9	10
100	ssms.exe	56164	56944	57348	57348	56916	56312	55504	35200	35324	35428
	sqlservr.exe	55036	55076	55116	55164	55204	55252	55304	55392	55432	55496
200	ssms.exe	37060	34896	35164	35796	55264	56096	56080	56884	57056	57056
	sqlservr.exe	55688	55772	55864	55964	56060	56140	56204	56292	56380	56500
300	ssms.exe	55920	55932	54872	55108	55220	54192	54204	36124	36024	36096
	sqlservr.exe	56604	56740	56764	56884	57004	57132	57344	57484	57596	57712
400	ssms.exe	57380	57380	55152	55312	55412	54508	54508	34448	34756	35680
	sqlservr.exe	57848	58004	58160	58320	58552	58728	58908	59076	59412	59432
500	ssms.exe	41088	38428	34684	34652	35220	54552	54444	34836	35308	35476
	sqlservr.exe	59644	59884	60036	60276	60420	60508	60572	60660	60748	60844
800	ssms.exe	40984	34300	35192	35072	35224	54404	35716	35728	36632	36092
	sqlservr.exe	61036	61336	61308	61500	61616	61768	61904	61952	62008	62096
1000	ssms.exe	56544	57844	57948	58040	57804	54436	55744	56368	56356	56004
	sqlservr.exe	62160	62204	62356	62496	62624	62740	62788	63012	62924	62996
1500	ssms.exe	55448	55396	55272	54984	54964	55384	36676	36196	36304	36368
	sqlservr.exe	63124	63180	63428	63380	63412	63412	63512	63432	63432	63432
2000	ssms.exe	58392	55116	55300	55676	55208	55204	56288	57000	57076	57132
	sqlservr.exe	63512	63536	63536	63536	63536	63648	63684	63592	63664	63644
3000	ssms.exe	54600	55624	55248	55336	55336	56200	36088	36252	36340	36316
	sqlservr.exe	63708	63772	63948	63960	63800	63840	63844	63844	64004	63844
5000	ssms.exe	54652	36308	36996	37444	58092	56332	56268	56288	56492	56800
	sqlservr.exe	64080	64980	65072	65076	65088	65084	65096	65088	65092	65008
10000	ssms.exe	57812	58032	57956	56196	37300	38848	38864	37232	37596	38244
	sqlservr.exe	65244	65544	65604	65888	66328	66756	66756	66772	66760	66756

Zdroj: vlastní

Hodnoty byly dále zpracovány, byl vypočten aritmetický průměr pro jednotlivé procesy a součtem hodnot obou procesů bylo určeno celkové využití paměti. Pro jednotlivé procesy byla vypočtena směrodatná odchylka, viz tabulka 20.

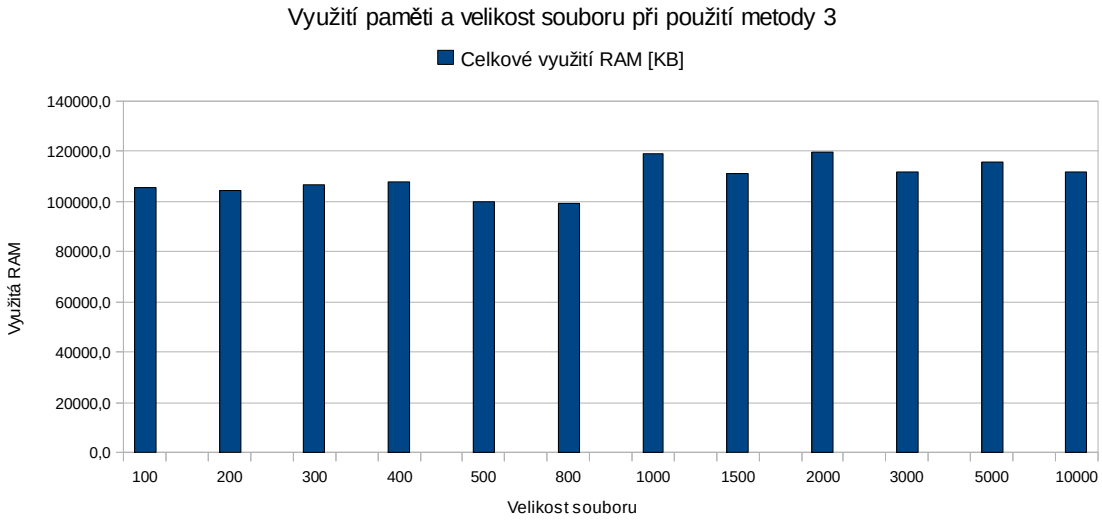
Tab. 20 Hodnoty odvozené z tabulky 19

Velikost souboru [řádky]	Proces	Průměrné využití RAM [KB]	Směrodatná odchylka	Směrodatná odchylka v %	Celkové využití RAM [KB]
100	ssms.exe	50248,80	10318,89	20,54	105496,00
	sqlservr.exe	55247,20	156,41	0,28	
200	ssms.exe	48135,20	10705,46	22,24	104221,60
	sqlservr.exe	56086,40	266,45	0,48	
300	ssms.exe	49369,20	9187,99	18,61	106495,60
	sqlservr.exe	57126,40	389,59	0,68	
400	ssms.exe	49453,60	10055,09	20,33	108097,60
	sqlservr.exe	58644,00	564,22	0,96	
500	ssms.exe	39868,80	7977,26	20,01	100228,00
	sqlservr.exe	60359,20	393,91	0,65	
800	ssms.exe	37934,40	6070,85	16,00	99586,80
	sqlservr.exe	61652,40	351,64	0,57	
1000	ssms.exe	56708,80	1185,88	2,09	119338,80
	sqlservr.exe	62630,00	316,02	0,50	
1500	ssms.exe	47699,20	9738,90	20,42	111073,60
	sqlservr.exe	63374,40	122,52	0,19	
2000	ssms.exe	56239,20	1120,96	1,99	119828,00
	sqlservr.exe	63588,80	65,26	0,10	
3000	ssms.exe	47734,00	9892,58	20,72	111590,40
	sqlservr.exe	63856,40	90,66	0,14	
5000	ssms.exe	50567,20	9460,09	18,71	115533,60
	sqlservr.exe	64966,40	313,89	0,48	
10000	ssms.exe	45808,00	10089,89	22,03	112048,80
	sqlservr.exe	66240,80	611,44	0,92	

Zdroj: vlastní

Jednotlivá pozorování procesu *ssms.exe* se vyznačují vyšší mírou variability, u procesu *sqlservr.exe* je velmi nízká míra variability.

Graf 7 představuje průběh sledované veličiny pro jednotlivé velikosti souborů.



Graf 7 Závislost využití paměti RAM a velikosti souboru při využití metody 3

Zdroj: vlastní

SQL Server spotřeboval relativně stálé množství paměti pro operaci se sledovanými soubory.

5.1.5 Porovnání jednotlivých metod

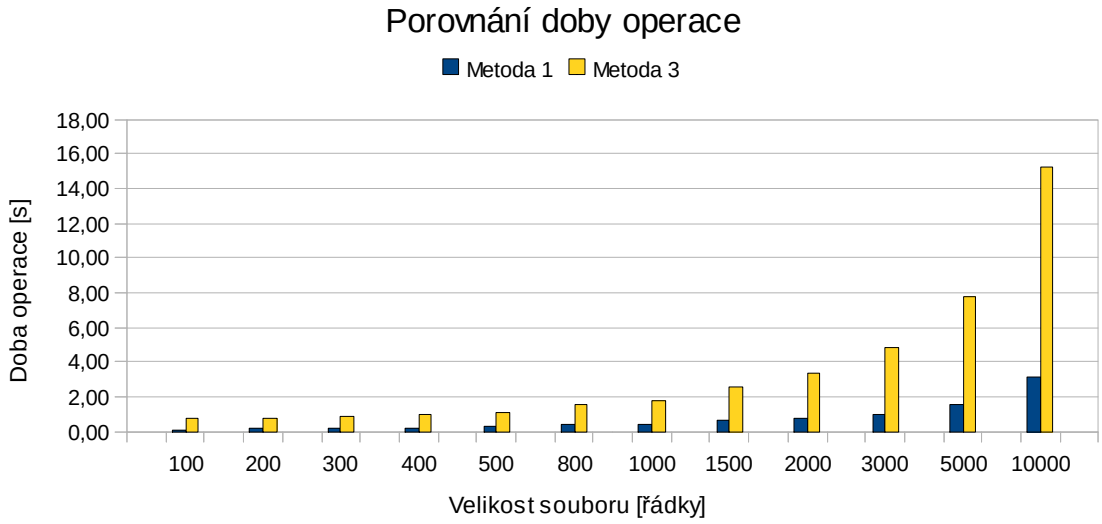
Pro jednotlivé metody byly vzájemně porovnávány hodnoty odvozené z naměřených hodnot, konkrétně se jedná o průměrné hodnoty doby operace, využití procesoru a RAM pro příslušné velikosti souborů, viz tabulka 21.

Tab. 21 Porovnání doby operace pro jednotlivé metody

Počet řádků souboru	Metoda 1	Metoda 2	Metoda 3
100	0,12	2,78	0,72
200	0,14	9,69	0,73
300	0,19	21,04	0,84
400	0,22	36,36	0,96
500	0,25	56,28	1,1
800	0,36	141,63	1,53
1000	0,40	220,07	1,82
1500	0,58	neměřeno	2,59
2000	0,69	neměřeno	3,31
3000	1,02	neměřeno	4,82
5000	1,57	neměřeno	7,77
10000	3,12	neměřeno	15,28

Zdroj: vlastní

Graf 8 představuje porovnání sledovaných veličin pro jednotlivé metody.



Graf 8 Porovnání doby operace pro jednotlivé metody

Zdroj: vlastní

Metoda 2 nebyla graficky porovnávána, jelikož doba importu dat touto metodou dosahuje vysokých hodnot a při zachování měřítka nelze graficky porovnat metodu 1 a 3. Vzhledem k neúnosným dobám operace 2 bylo prováděno měření pouze do velikosti souboru 1000 řádků. Import dat proběhl nejrychleji při použití metody 1.

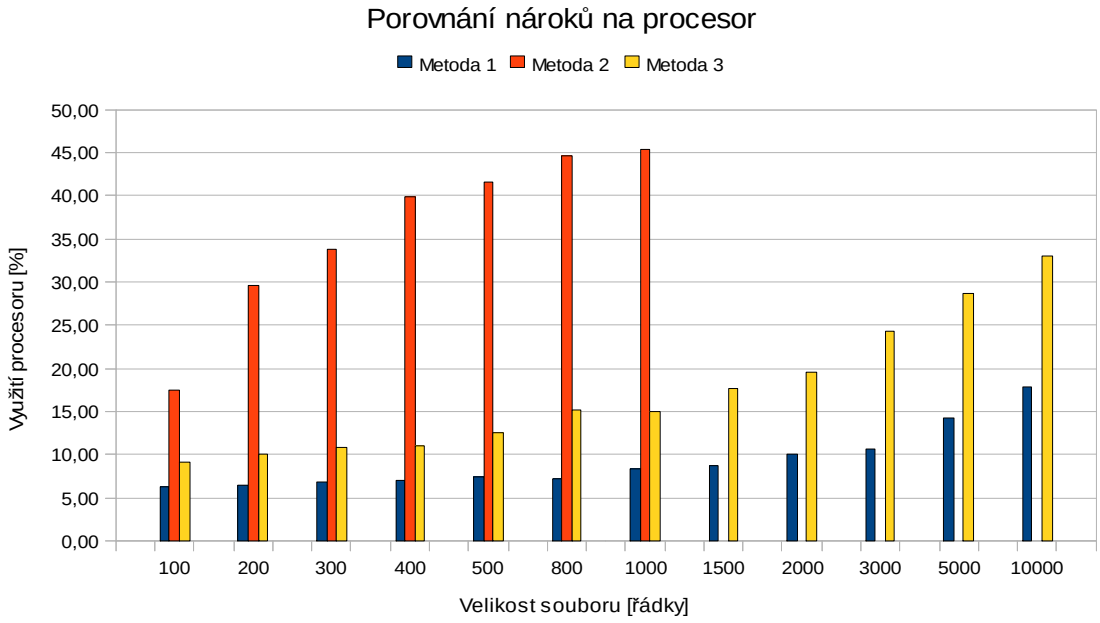
Importování největšího souboru do SQL Serveru 2005 bylo rychlejší než u SQL Serveru 2008 a nedosáhlo doby importu souboru o 1000 řádcích do SQL Serveru 2008 – doby 220,07 s.

Tab. 22 Porovnání nároků jednotlivých metod na procesor

Počet řádků souboru	Metoda 1	Metoda 2	Metoda 3
100	6,18	17,37	9,08
200	6,38	29,67	10,13
300	6,77	33,83	10,85
400	6,93	39,92	11,05
500	7,29	41,56	12,57
800	7,21	44,63	15,21
1000	8,34	45,30	14,99
1500	8,76	neměřeno	17,66
2000	10,05	neměřeno	19,57
3000	10,62	neměřeno	24,33
5000	14,14	neměřeno	28,59
10000	17,87	neměřeno	32,99

Zdroj: vlastní

Graf 9 představuje porovnání sledovaných veličin pro jednotlivé metody.



Graf 9 Porovnání nároků jednotlivých metod na procesor

Zdroj: vlastní

Použití metody 1 představovalo menší nároky na procesor, než použití metody 3.

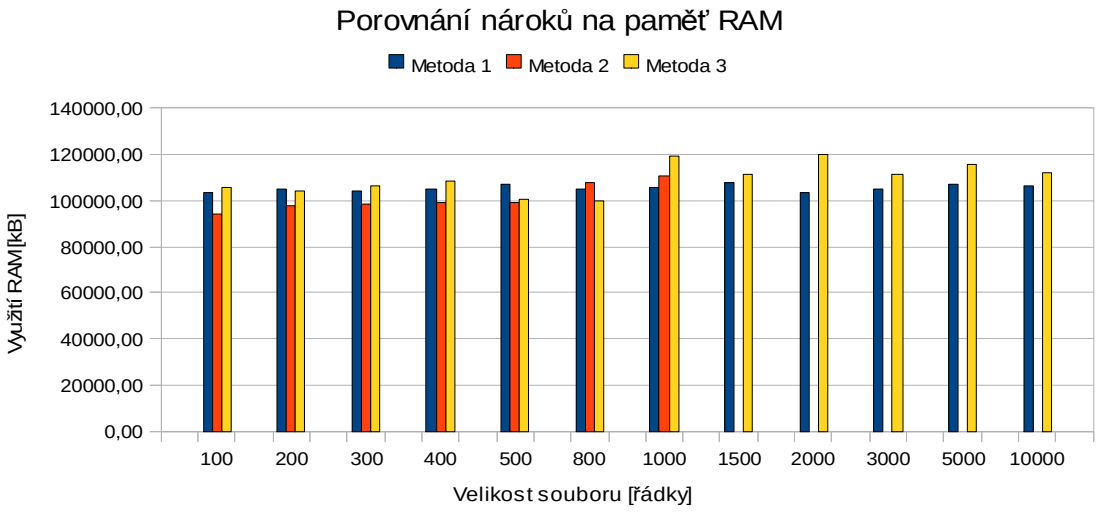
Nároky na procesor byly u SQL Serveru 2005 přibližně stejné, nárůst hodnot se zastavil na 50%, tuto hodnotu výrazně nepřekračoval a zůstal na této úrovni po zbytek měření.

Tab. 23 Porovnání nároků jednotlivých metod na paměť RAM

Počet řádků souboru	Metoda 1	Metoda 2	Metoda 3
100	103355,6	93862	105496
200	105039,2	97971,6	104221,6
300	103918,8	98708	106495,6
400	105098	99148	108097,6
500	106820,4	98863,2	100228
800	104880,8	107703,6	99586,8
1000	105447,6	110805,6	119338,8
1500	107465,2	neměřeno	111073,6
2000	103130	neměřeno	119828
3000	105010,8	neměřeno	111590,4
5000	106680,8	neměřeno	115533,6
10000	105981,6	neměřeno	112048,8

Zdroj: vlastní

Graf 10 představuje porovnání sledovaných veličin pro jednotlivé metody.



Graf 10 Porovnání nároků jednotlivých metod na paměť RAM

Zdroj: vlastní

U jednotlivých metod se nevyskytují tak velké rozdíly ve využití paměti RAM, jako u zbylých sledovaných parametrů.

U SQL Serveru 2005 docházelo k přibližně stejnému využití paměti RAM, jako u SQL Serveru 2008, provedené pokusy naznačují, že všechny tři metody spotřebují přibližně stejné množství paměti RAM.

5.1.6 Doporučení pro SQL Server 2008

Vzhledem k naměřeným hodnotám a závěrům plynoucím ze začátku kapitoly 5, je nejefektivnější cestou k zpracování XML dat na SQL Serveru 2008 využití dat uložených jako atributy zpracovaných metodou 1.

5.2 Nativní XML databáze

Pro získání reprezentativních výsledků bylo rozhodnuto provést stejný test, jako pro SQL Server 2008 i pro nativní XML databázi. Bylo vybíráno z open source databází uvedených v [37], konkrétně se jednalo o dbXML – 2.0, eXist 1.4, Sedna 3.2.91, BaseX 5.7 a Apache Xindice 1.1. Výběr databáze, na které se bude zkoušet práce s elementy a atributy proběhl ve dvou fázích. Prvním vyřazovacím kritériem byla přítomnost spouštěcího souboru umožňujícího okamžitou práci s databází, což splnily databáze BaseX a eXist. Z těchto dvou kandidátů byla po letmém prozkoumání chodu a s přihlédnutí k vypracované dokumentaci zvolena databáze eXist.

5.2.1 eXist

Problematika uložení dat jako elementy a atributy byla dále zkoumána na nativní open source XML databázi eXist.

5.2.2 Metodika měření

Byl použitý stejný hardware a soubory jako při testu v kapitole 5.1.1.

Při měření byly spuštěny pouze programy nutné k provedení testu, a to:

- eXist Client Shell
- eXist Admin Client
- Sledování spolehlivosti a výkonu
- Průzkumník Windows

Software počítače byl doplněn o databázi eXist, Java Runtime Environment 1.6 a JDK 6. Sledovaným procesem byl *java.exe*. Jednotlivé soubory byly desetkrát importovány, vzniklá tabulka byla po každém importu smazána. U jednotlivých měření nebyla zaznamenávána doba operace z důvodu absence této funkce v databázi eXist; největší soubory byly načteny do 7 sekund, načítání dat uložených jako elementy bylo delší než načítání dat uložených jako atributy. Zkoumání bylo zaměřeno pouze na chování aplikace při práci s XML daty ve formě atributů a elementů. Import dat probíhal aktivací ikon v ovládacím panelu databáze a ručním výběrem požadovaného souboru. Dvě následující kapitoly obsahují naměřená data, rozbor výsledků následuje samostatně.

5.2.3 Data uložená jako atributy

Data byla manuálně vybrána v menu databáze a byl proveden import tabulky. Výsledky jsou v tabulce 24. Naměřené hodnoty představují procentuální podíl na celkovém výkonu procesoru.

Tab. 24 Naměřené hodnoty při importu atributů

Velikost souboru [řádky]	Proces	Měření									
		1	2	3	4	5	6	7	8	9	10
100	java.exe	1,5	1,5	2,4	1,7	2,2	3,3	2,0	2,6	2,4	2,0
200	java.exe	2,0	1,5	2,1	2,1	1,8	2,7	2,4	1,7	3,5	3,0
300	java.exe	1,8	2,1	2,0	1,8	3,6	1,6	2,6	2,5	3,0	2,8
400	java.exe	2,8	2,0	1,7	3,4	2,6	2,8	2,5	2,7	2,6	2,8
500	java.exe	2,2	4,0	2,6	3,0	2,8	2,1	4,3	3,4	3,7	2,8
800	java.exe	3,0	3,9	2,5	3,1	3,4	3,2	3,3	2,6	3,1	4,4
1000	java.exe	4,4	3,3	2,9	5,0	3,2	3,8	3,8	4,2	4,4	2,8
1500	java.exe	3,9	3,4	3,5	4,3	3,7	3,6	3,5	3,8	3,7	4,8
2000	java.exe	5,5	6,1	5,1	5,5	5,8	6,1	3,9	5,8	4,9	4,8
3000	java.exe	5,9	6,7	7,0	5,5	5,8	6,8	6,3	7,3	6,4	6,8
5000	java.exe	6,7	7,7	5,2	10,6	9,7	7,2	10,4	8,4	9,1	8,3
10000	java.exe	10,6	12,8	13,5	13,7	13,3	13,6	16,1	13,9	12,2	13,0

Zdroj: vlastní

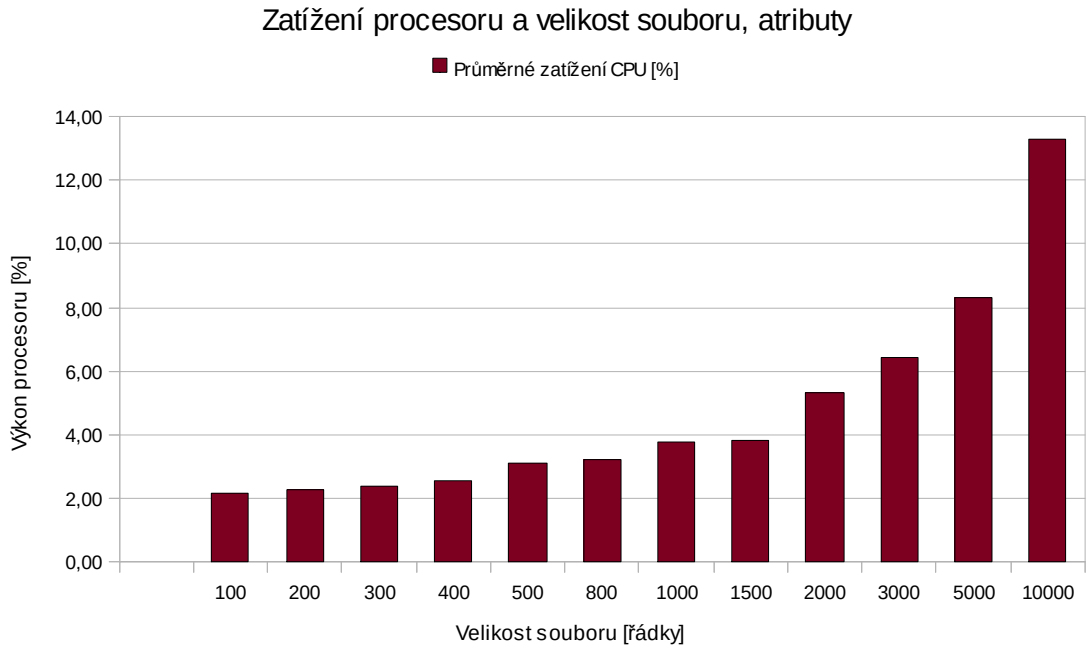
Hodnoty byly dále zpracovány, byl vypočten aritmetický průměr a směrodatná odchylka, viz tabulka 25.

Tab. 25 Hodnoty odvozené z tabulky 24

Velikost souboru [řádky]	Proces	Průměrné zatížení CPU [%]	Směrodatná odchylka	Směrodatná odchylka v %
100	java.exe	2,16	0,55	25,56
200	java.exe	2,28	0,63	27,41
300	java.exe	2,38	0,63	26,63
400	java.exe	2,59	0,47	17,97
500	java.exe	3,09	0,74	24,00
800	java.exe	3,25	0,56	17,36
1000	java.exe	3,78	0,73	19,19
1500	java.exe	3,82	0,43	11,23
2000	java.exe	5,35	0,69	12,84
3000	java.exe	6,45	0,58	8,93
5000	java.exe	8,33	1,70	20,42
10000	java.exe	13,27	1,39	10,45

Zdroj: vlastní

Jednotlivá pozorování se vyznačují proměnlivou mírou variability, které klesá s rostoucí velikostí souboru. Graf 11 představuje průběh sledované veličiny pro jednotlivé velikosti souborů.



Graf 11 Závislost využití procesoru a velikosti souboru při importu dat

Zdroj: vlastní

Dále bylo sledováno, jak je využívána paměť, konkrétně jak velká je soukromá pracovní sada procesu *java.exe*. Naměřené hodnoty jsou v tabulce 26. Jednotlivé hodnoty jsou v KB.

Tab. 26 Využití paměti, data uložena jako atributy

Velikost souboru [řádky]	Proces	Měření									
		1	2	3	4	5	6	7	8	9	10
100	java.exe	96324	96432	96540	98276	99016	99116	99140	99272	99452	99460
200	java.exe	99632	99716	99812	99816	99824	99848	99908	100036	100176	100276
300	java.exe	101044	101276	101284	101312	101344	101384	101412	101492	101548	101560
400	java.exe	102940	102968	103104	103108	103116	103228	103260	103432	103460	103496
500	java.exe	104644	104704	104708	104728	104744	104744	104792	104832	104988	105128
800	java.exe	106504	106540	106732	106784	106804	106936	106968	106952	107072	107128
1000	java.exe	102504	102192	102128	100172	98956	99336	99548	99604	99976	100152
1500	java.exe	107804	108104	109012	109240	109592	110352	110448	110768	111500	112168
2000	java.exe	132804	133832	135712	137528	139308	139912	141304	142552	143556	144484
3000	java.exe	171904	171884	171888	171888	171924	171900	171924	171900	171984	171988
5000	java.exe	171968	171984	171992	172016	173556	173328	173392	173392	173392	173396
10000	java.exe	171496	171552	171548	171532	171552	171540	171576	171660	171680	171780

Zdroj: vlastní

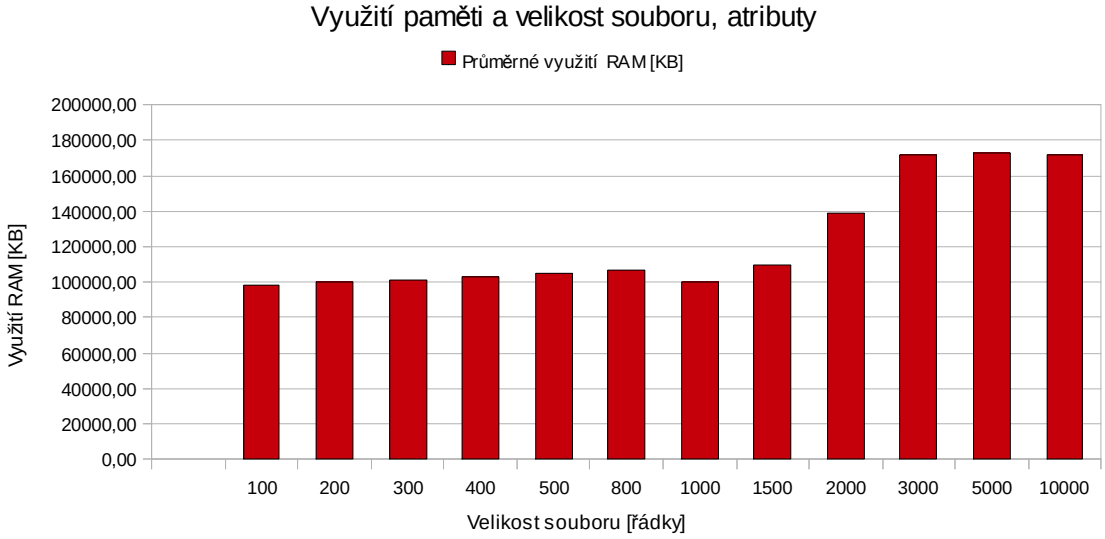
Hodnoty byly dále zpracovány, byl vypočten aritmetický průměr a směrodatná odchylka, viz tabulka 27.

Tab. 27 Hodnoty odvození z tabulky 26

Velikost souboru [řádky]	Proces	Průměrné využití RAM [KB]	Směrodatná odchylka	Směrodatná odchylka v %
100	java.exe	98302,80	1333,07	1,36
200	java.exe	99904,40	201,35	0,20
300	java.exe	101365,60	153,11	0,15
400	java.exe	103211,20	199,54	0,19
500	java.exe	104801,20	148,13	0,14
800	java.exe	106842,00	209,20	0,20
1000	java.exe	100456,80	1310,72	1,30
1500	java.exe	109898,80	1410,3	1,28
2000	java.exe	139099,20	4050,5	2,91
3000	java.exe	171918,4	38,2	0,02
5000	java.exe	172841,6	735,2	0,43
10000	java.exe	171591,6	87,3	0,05

Zdroj: vlastní

Naměřené hodnoty vykazují velmi nízkou variabilitu. Graf 12 představuje průběh sledované veličiny pro jednotlivé velikosti souborů.



Graf 12 Závislost využití paměti RAM a velikosti souboru, elementy

Zdroj: vlastní

5.2.4 Data uložená jako elementy

Data byla manuálně vybrána v menu databáze a byl proveden import tabulky. Výsledky jsou v tabulce 28. Naměřené hodnoty představují procentuální podíl na celkovém výkonu procesoru.

Tab. 28 Naměřené hodnoty při importu elementů

Velikost souboru [řádky]	Proces	Měření									
		1	2	3	4	5	6	7	8	9	10
100	java.exe	1,4	2,5	3,1	2,4	2,9	2,3	1,9	2,8	3,0	1,9
200	java.exe	2,4	2,8	3,6	1,7	1,8	1,8	3,4	3,2	2,3	2,9
300	java.exe	3,8	2,6	2,2	2,6	4,2	3,9	4,0	2,9	2,2	4,2
400	java.exe	2,6	2,5	4,4	2,9	4,4	3,2	2,4	2,6	2,5	2,1
500	java.exe	3,2	3,0	3,6	4,5	5,2	4,4	3,6	4,3	3,6	3,3
800	java.exe	4,1	5,4	4,0	4,3	4,4	5,4	4,7	5,9	3,8	6,9
1000	java.exe	4,3	4,9	6,1	6,5	6,0	5,1	6,0	6,1	7,4	6,5
1500	java.exe	5,2	5,3	7,2	7,3	6,3	6,8	6,6	7,0	7,2	6,4
2000	java.exe	7,0	9,6	8,1	6,0	7,2	8,1	7,7	8,1	7,0	6,6
3000	java.exe	9,1	7,9	8,1	10,1	7,8	10,3	9,6	9,4	8,9	9,9
5000	java.exe	13,2	15,1	13,5	12,2	13,0	15,9	13,3	14,2	14,0	12,1
10000	java.exe	20,7	20,6	21,0	19,2	19,7	24,4	19,6	17,4	17,4	18,2

Zdroj: vlastní

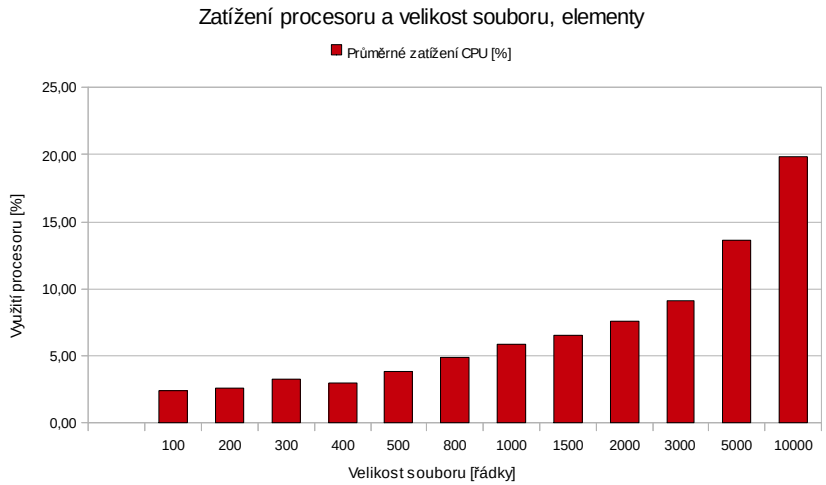
Hodnoty byly dále zpracovány, byl vypočten aritmetický průměr a směrodatná odchylka, viz tabulka 29.

Tab. 29 Hodnoty odvozené z tabulky 28

Velikost souboru [řádky]	Proces	Průměrné zatížení CPU [%]	Směrodatná odchylka	Směrodatná odchylka v %
100	java.exe	2,42	0,56	22,95
200	java.exe	2,59	0,70	26,84
300	java.exe	3,26	0,83	25,59
400	java.exe	2,96	0,81	27,46
500	java.exe	3,87	0,70	18,03
800	java.exe	4,89	0,99	20,23
1000	java.exe	5,89	0,90	15,26
1500	java.exe	6,53	0,76	11,57
2000	java.exe	7,54	1,01	13,38
3000	java.exe	9,11	0,92	10,08
5000	java.exe	13,65	1,20	8,76
10000	java.exe	19,82	2,07	10,44

Zdroj: vlastní

Jednotlivá pozorování se vyznačují proměnlivou mírou variability, které klesá s rostoucí velikostí souboru. Graf 13 představuje průběh sledované veličiny pro jednotlivé velikosti souborů.



Graf 13 Závislost využití procesoru a velikosti souboru při importu dat

Zdroj: vlastní

Dále bylo sledováno, jak je využívána paměť, konkrétně jak velká je soukromá pracovní sada procesu *java.exe*. Naměřené hodnoty jsou v tabulce 30. Jednotlivé hodnoty jsou v KB.

Tab. 30 Využití paměti, data uložena jako elementy

Velikost souboru [řádky]	Proces	Měření									
		1	2	3	4	5	6	7	8	9	10
100	java.exe	97156	97272	96804	97180	97240	97356	97360	97552	98140	98264
200	java.exe	100352	100484	100464	100572	100584	100740	100956	100956	100956	101052
300	java.exe	101716	101780	101816	102228	102340	102360	102338	102512	102636	102676
400	java.exe	103636	103876	103960	104044	104108	104184	104352	104388	104472	104488
500	java.exe	105260	105268	105436	105576	105668	105760	105840	106012	106120	106180
800	java.exe	106340	107576	107616	106952	106108	105356	105992	105948	106124	106616
1000	java.exe	97644	97556	98248	98604	100200	102244	103932	105552	107008	107584
1500	java.exe	114784	115760	117948	117812	119788	123380	127060	127980	129776	132772
2000	java.exe	143756	170136	170236	171144	172488	172488	172548	172072	171788	171804
3000	java.exe	172000	172016	172076	172076	172076	172080	172084	172084	172100	171968
5000	java.exe	173048	173056	172884	172280	173468	171680	171368	171368	171416	171776
10000	java.exe	171712	171720	171720	171800	171816	172272	172272	173360	172308	172300

Zdroj: vlastní

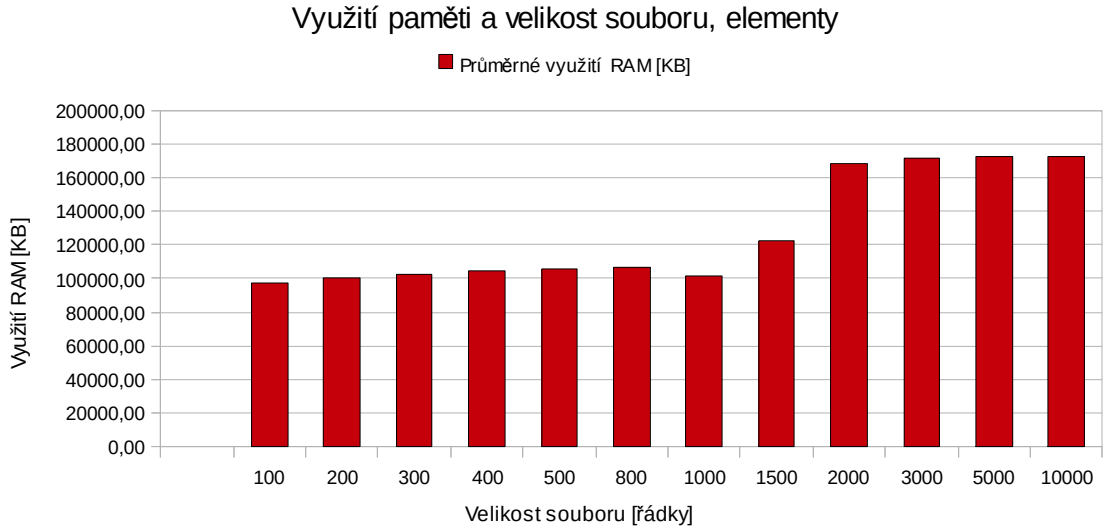
Hodnoty byly dále zpracovány, byl vypočten aritmetický průměr a směrodatná odchylka, viz tabulka 31.

Tab. 31 Hodnoty odvozené z tabulky 30

Velikost souboru [řádky]	Proces	Průměrné využití RAM [KB]	Směrodatná odchylka	Směrodatná odchylka v %
100	java.exe	97432,40	449,12	0,46
200	java.exe	100711,60	252,56	0,25
300	java.exe	102240,20	352,92	0,35
400	java.exe	104150,80	279,86	0,27
500	java.exe	105712,00	332,03	0,31
800	java.exe	106462,80	730,24	0,69
1000	java.exe	101857,20	3944,62	3,87
1500	java.exe	122706,00	6365,2	5,19
2000	java.exe	168846,00	8859,6	5,25
3000	java.exe	172056,0	44,4	0,03
5000	java.exe	172234,4	814,5	0,47
10000	java.exe	172128,0	509,1	0,30

Zdroj: vlastní

Naměřené hodnoty vykazují velmi nízkou variabilitu. Graf 14 představuje průběh sledované veličiny pro jednotlivé velikosti souborů.



Graf 14 Závislost využití paměti RAM a velikosti souboru, elementy

Zdroj: vlastní

5.2.5 Porovnání

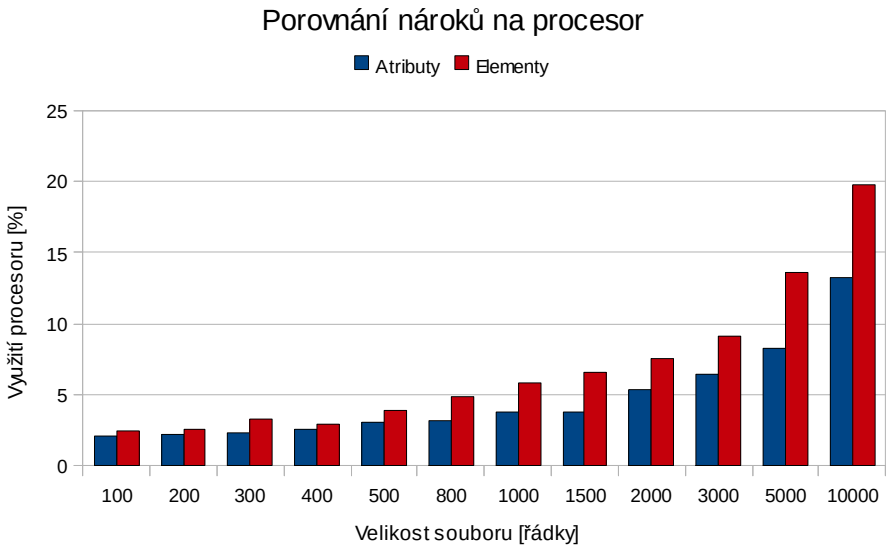
Využití procesoru v závislosti k způsobu ukládání dat je porovnána v tabulce 32.

Tab. 32 Porovnání nároků na procesor

Počet řádků souboru	Atributy	Elementy
100	2,16	2,42
200	2,28	2,59
300	2,38	3,26
400	2,59	2,96
500	3,09	3,87
800	3,25	4,89
1000	3,78	5,89
1500	3,82	6,53
2000	5,35	7,54
3000	6,45	9,11
5000	8,33	13,65
10000	13,27	19,82

Zdroj: vlastní

Graf 15 představuje porovnání sledovaných veličin pro jednotlivé přístupy



Graf 15 Porovnání nároků na procesor pro jednotlivé přístupy

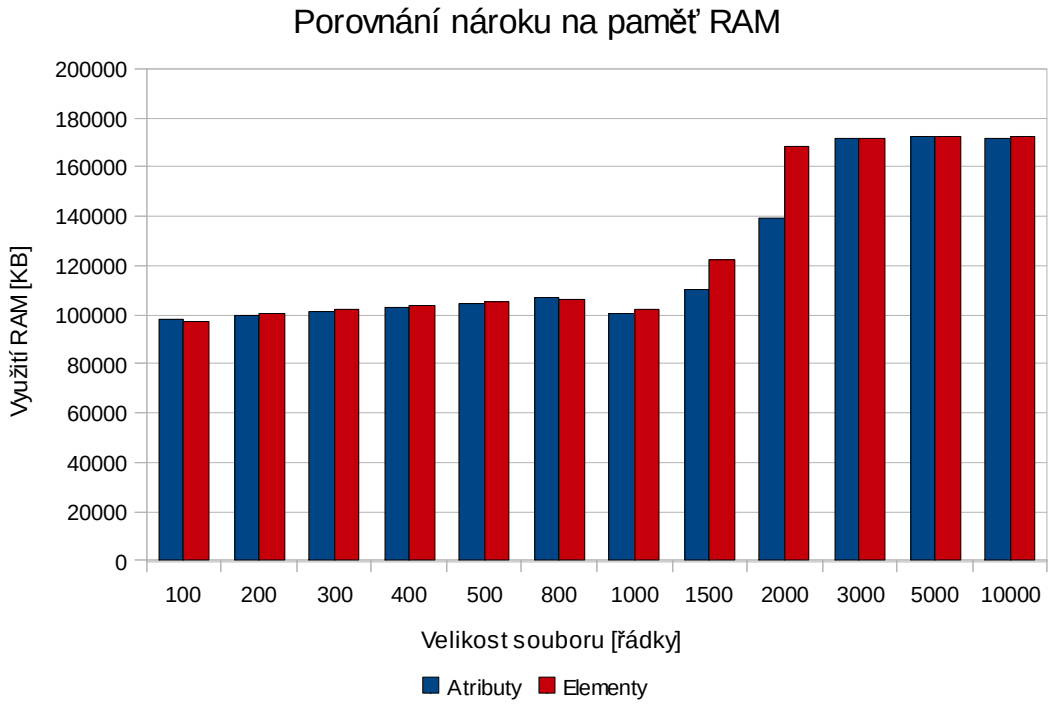
Zdroj: vlastní

Tab. 33 Porovnání nároků na paměť

Počet řádků souboru	Atributy	Elementy
100	98302,8	97432,4
200	99904,4	100711,6
300	101365,6	102240,2
400	103211,2	104150,8
500	104801,2	105712
800	106842	106462,8
1000	100456,8	101857,2
1500	109898,8	122706
2000	139099,2	168846
3000	171918,4	172056
5000	172841,6	172234,4
10000	171591,6	172128

Zdroj: vlastní

Graf 16 představuje porovnání sledovaných veličin pro jednotlivé přístupy



Graf 16 Porovnání nároků na paměť RAM pro jednotlivé přístupy

Zdroj: vlastní

5.3 Výsledek měření

Porovnání měření prokázalo, že import dat uložených jako atributy přináší menší nároky na procesor. Na rozdíl od SQL Serveru 2008 není u databáze eXist rozdíl ve spotřebované paměti pro import souborů uložených jako elementy a atributy. Stejně jako u SQL Serveru lze označit data uložená jako atributy za efektivnější způsob nakládání s XML, než data uložená jako elementy.

6 Závěr

Z podstaty XML pramení nepřeberné množství uživatelem specifikovaných aplikací, ale největší přínos je ve využití XML jako největšího společného dělitele mezi různými programy ke vzájemné výměně dat.

Výsledky zkoumání možností využití XML v kancelářských aplikacích se dají shrnout následovně: v komerčním softwaru Microsoft Office 2007 je integrace s XML zvládnuta lépe – nikoliv bezchybně, než u open source OpenOffice.

Cílem práce bylo prověřit možnosti volby formy XML souborů a poskytnout návod k efektivnímu zacházení s datově orientovanými dokumenty. Domnívám se, že se podařilo popsat na příkladech efektivní a neefektivní způsoby nakládání s XML soubory pro SQL Server 2008. Pokusy s nativní XML databází vedly ke stejným závěrům jako u relační databáze.

Volba nativní XML databáze, na které měly být odzkoušeny jednotlivé přístupy, byla poznamenána řadou nefunkčních zástupců ze skupiny volně šiřitelného software, zejména se jednalo o absenci verzí pracujících pod operačním systémem Windows Vista a absenci spouštěcího souboru, který by umožnil snadný start.

Ekonomický přínos efektivního využití XML se liší od konkrétní situace a nelze obecně vyčíslit. Urychlení integrace XML s relační databází pro situace, kdy jsou tyto operace naplánovány na dobu nulové aktivity uživatelů a je dostatek času pro jejich provedení, přinese úsporu pouze v ušetřené spotřebě elektrické energie a nižšího opotřebení hardware. Zefektivnění práce s XML v nepřetržitě používaném systému může přinést nemalé úspory v nákladech na hardware a snížení celkové doby operace, od čehož se odvíjí rychlost výstupu pro koncového uživatele.

Práce je zaměřena na malou část problematiky databází a to na integraci XML obsahu do SQL Serveru a nativní databáze eXist, dalo by se navázat popsáním efektivního nakládání

s XML v této práci neodzkoušených a v praxi používaných databázových systémech, jako například Oracle, Tamino, doplnění dokumentace k těmto programům o výsledky a doporučení a demonstraci vhodných a nevhodných způsobů používání XML. Lze předpokládat, že programovací jazyky budou stále složitější a bude obtížnější jim do detailu porozumět, proto by měla kvalitní dokumentace obsahovat příklady efektivních a neefektivních postupů.

7 Seznam použité literatury

Citace

[1] BOLETTIERI, P., FALCHI, F., et al. Automatic Metadata Extraction and Indexing for Reusing e-Learning Multimedia Objects. *Workshop on multimedia information retrieval on The many faces of multimedia semantics.* 2007, pg. 21-28, ISBN 978-1-59593-782-7.

[2] SPENCER, J. Using XML to map relationships in hacker forums. *Proceedings of the 46th Annual Southeast Regional Conference on XX.* 2008, pg. 487-489, ISBN 978-1-60558-105-7.

[3] SKIBIŃSKI, P., GRABOWSKI, S., SWACHA, J. Effective asymmetric XML Compression. *Software: Practice and Experience*. 2007, vol. 38, issue 10, pg. 1027-1047. [online]. [cit. 2009-11-18]. Dostupný z WWW: <http://www3.interscience.wiley.com/cgi-bin/fulltext/116839064/PDFSTART>

[4] KAPPEL, G., KAPSAMMER, E., RETSCHITZEGGER, W. Integrating XML and Relational Database Systems. [online]. [cit. 2009-11-18]. Dostupný z WWW: <http://www.springerlink.com/content/tv41677322272431/fulltext.pdf>

[5] JENNINGS, R., Special Edition Using Microsoft Office Access 2007. 1st ed., Que Publishing, 2007. ISBN 0-7897-3597-0.

[6] MACDONALD, M., *Excel 2007 The Missing Manual*. 1st ed., O'Reilly Media, 2007. ISBN 0-596-52759-4.

[7] BALMIN, A., PAPAKONSTANTINOU, Y. Storing and querying XML data using denormalized relational databases. *The VLDB Journal.* 2005, vol. 14, issue 1, pg. 30-49. ISSN 1066-8888.

[8] Papers about XML [online]. [cit. 2009-11-17]. Dostupný z WWW: <http://www.rpbourret.com/xml>

[9] SALMINEN, A., WM. TOMPA, F. Requirements for XML Document Database Systems. *Proceedings of the 2001 ACM Symposium on Document engineering table of contents.* 2001, pg. 85-94. ISBN 1-58113-432-0.

[10] Date and Time Formats [online]. [cit. 2009-11-14]. Dostupný z WWW: <http://www.w3.org/TR/NOTE-datetime>

[11] GRABS, T., BÖHM, K., SCHEK, H. XMLTM: efficient transaction management for XML documents. *Proceedings of the eleventh international conference on Information and knowledge management.* 2002, pg. 142-152. ISBN 1-58113-492-4.

[12] HEAD, M., et al. Benchmarking XML Processors for Applications in Grid Web Services. *Proceedings of the 2006 ACM/IEEE conference on Supercomputing.* 2006. ISBN 0-7695-2700-0.

[13] XHTML 1.0 The Extensible HyperText Markup Language (Second Edition) [online]. [cit. 2009-11-29]. Dostupný z WWW: <http://www.w3.org/TR/xhtml1>

[14] DocBook [online]. [cit. 2009-11-29]. Dostupný z WWW: <http://www.kosek.cz/xml/db>

[15] rpbourret.com - XML and Databases [online]. [cit. 2009-7-31]. Dostupný z WWW: <http://www.rpbourret.com/xml/XMLAndDatabases.htm>

[16] Vyhláška o stanovení podrobností užívání a provozování informačního systému datových schránek [online]. [cit. 2009-11-30]. Dostupný z WWW: <portal.justice.cz/justice2/soubor.aspx?id=81262>

[17] KML Tutorial [online]. [cit. 2009-11-30]. Dostupný z WWW: <http://code.google.com/intl/cs/apis/kml/documentation/kml_tut.html>

Bibliografie

[18] JELEN, B., *Special Edition Using Microsoft Office Excel 2007.* 1st ed., Que Publishing, 2006. ISBN 0-7897-3611-X.

[19] GRAFF, J., Weinberg, P., *SQL Kompletní průvodce.* CP Books, 2005. ISBN 80-251-0369-2.

[20] ROSMAITA, B. Accessibility First! A New Approach to Web Design. *Proceedings of the 37th SIGCSE technical symposium on Computer science education.* 2006, pg. 270-274. ISBN 1-59593-259-3.

[21] MLÝNKOVÁ, I., aj. *Technologie XML.* 1. Vyd. Praha: Karolinum, 2006. ISBN 80-246-1272-0.

[22] POKORNÝ, J., aj. XML technologie. 1. Vyd. Praha: Grada, 2008. ISBN 978-80-247-2725-7.

[23] KML [online]. [cit. 2009-11-30]. Dostupný z WWW: <http://www.opengeospatial.org/standards/kml>

[24] Zákony - Vyhledávání - Portál veřejné správy České republiky [online]. [cit. 2009-11-30]. Dostupný z WWW: <http://portal.gov.cz/wps/portal/_s.155/701?kam=zakon&c=300/2008>

[25] Právní předpisy k datovým schránkám [online]. [cit. 2009-11-30]. Dostupný z WWW: <http://www.mvcr.cz/clanek/navrhy-provadecich-pravnich-predpisu-k-datovym-schrankam.aspx>

[26] KML Documentation Introduction [online]. [cit. 2009-11-30]. Dostupný z WWW: <http://code.google.com/intl/cs/apis/kml/documentation>

[27] HTML 5 Reference [online]. [cit. 2009-11-29]. Dostupný z WWW: <http://dev.w3.org/html5/html-author>

[28] HTML 5 differences from HTML 4 [online]. [cit. 2009-11-29]. Dostupný z WWW: <http://www.w3.org/TR/html5-diff>

[29] DocBook 5.0 [online]. [cit. 2009-11-29]. Dostupný z WWW: <http://www.docbook.org/tdg5/en/html/docbook.html>

[30] DocBook V5.0 The Transition Guide [online]. [cit. 2009-11-29]. Dostupný z WWW: <http://www.docbook.org/docs/howto/2006-10-22/howto.pdf>

[31] eXist-db Open Source Native XML Database [online]. [cit. 2009-12-1]. Dostupný z WWW: <http://exist-db.org>

[32] Tarmac CZ a.s. [online]. [cit. 2009-12-4]. Dostupný z WWW: <http://www.tarmac.cz>

[33] Apache Xindice [online]. [cit. 2009-12-11]. Dostupný z WWW: <http://xml.apache.org/xindice>

[34] http://basex,org/ [online]. [cit. 2009-12-11]. Dostupný z WWW: <http://basex.org>

[35] Sedna XML Database [online]. [cit. 2009-12-11]. Dostupný z WWW: <http://modis.ispras.ru/sedna>

[36] VANICKÝ, M., XML a databáze. [Diplomová práce]. Praha: Vysoká škola ekonomická v Praze – Fakulta informatiky a statistiky, 2004.

[37] Quick Start Guide [online]. [cit. 2009-12-17].
Dostupný z WWW: <http://exist.sourceforge.net/quickstart.html>

[38] rpbourret.com – XML Database Products [online]. [cit. 2009-18-12].
Dostupný z WWW:
<http://www.rpbourret.com/xml/XMLDatabaseProds.htm#native>