

Seminarausarbeitung

Effektives Management von RDF Modellen am Beispiel von RDFCore

Ronja Düffel

betreut von
Dipl. Math. Karsten Tolle

31. Januar 2004

1 Einführung

Die rasante Entwicklung des Internets in den letzten 10 Jahren, hat dazu geführt, dass heute jede nur denkbare Information im World Wide Web verfügbar ist, ...man muss sie „nur“ finden. Dazu ist es dringend notwendig diese riesige Menge an Informationen effektiv zu verwalten, denn niemand will stundenlang auf ein Suchergebnis warten, in dem die gewünschte Information dann immer noch nicht zu finden ist.

Suchmaschinen, wie Altavista, Infoseek und Excite arbeiten mit Brute Force Methoden (vergl. [1]) und liefern meist wenig zufriedenstellende Ergebnisse. Sie durchwandern das Netz, lesen Dokumente Wort für Wort und „finden“ so Informationen. Dieses Vorgehen ist vergleichbar mit dem durchlesen einer Bibliothek Buch für Buch, um Informationen über z.B. Rentierhaltung zu finden. Bibliotheken besitzen glücklicherweise ein sehr viel effizienteres System um Informationen zu finden, den Katalog. In ihm sind Daten über jedes Buch gespeichert, wie Autor, Inhalt, ISBN, und vor allem der Standort. So ist es möglich die Bücher nach Autor ect... auszugeben und Informationen nachzuschlagen.

Ähnlicher Hilfsmittel bedienen wir uns auch, wenn wir eine CD bei HMV oder die Telefonnummer eines Klempners in den Gelben Seiten suchen. Wir schlagen Informationen nach anhand von Metadaten, Informationen über Informationen. Und obwohl HMV, eine Bibliothek und die Gelben Seiten sehr unterschiedliche Metadaten verwenden, - Autor, Interpret, Branche,... - , sind die Operationen auf diesen doch sehr ähnlich.

Das Web ist vergleichbar mit einer RIESIG großen Bibliothek, in der es ausser Büchern aber auch alle möglichen anderen Dinge gibt (CDs, Klempner, ect...). Dennoch benötigt das Web dringend Metadaten, um eine effektive Nutzung zu ermöglichen. Diese Idee verbirgt sich hinter dem Namen *Sematisches Web*. Um Metadaten im World Wide Web zu realisieren wurde RDF entwickelt. Seit 1999 hat es den Status einer Recommendation des W3C [1].

2 RDF

RDF steht für Resource Description Framework, eine Sprache die entwickelt wurde um Metadaten im World Wide Web zu speichern und auszutauschen. Durch die Struktur des World Wide Web ist die Einführung von Metadaten jedoch mit einigen Problemen verbunden.

- Zum einen enthält das Web nicht nur Bücher, sondern eine Vielzahl von Informationen nach denen gesucht wird. Dementsprechend vielfältig sind die Metadaten, und dementsprechend vielfältig müssen auch die Beschreibungsmöglichkeiten der Metadaten sein.
- Ferner ist das Netz „nicht direktiv“. Jeder kann seine Informationen ins Netz stellen, und seine eigenen Metadaten entwerfen. Es gibt keinen Standard und niemanden, der einen solchen durchsetzen und kontrollieren könnte.
- Desweiteren müssen Metadaten zwischen Anwendungen austauschbar sein, so dass Anwendungen auch auf Informationen zugreifen können die ursprünglich nicht für sie erstellt wurden.

RDF versucht all diesen Anforderungen gerecht zu werden. Es bietet einen einfachen Weg Aussagen über Web-Ressourcen zu machen [5], wobei mit Ressourcen alles gemeint ist, was eine URI (Uniform Resource Identifier) haben kann. Also alles, was im Web identifiziert werden kann, sowohl Web Dokumente als auch einzelne Elemente eines XML Dokuments [1].

2.1 RDF-Graphen

Aussagen über Web-Ressourcen werden als Tripel (Statements) dargestellt, die aus Subjekt, Prädikat und Objekt bestehen. Man sagt, <Subjekt> hat eine Eigenschaft <Prädikat> mit Wert <Objekt> (vergl. [2]). RDF verwendet URI references (URIref, URI#fragment identifier) um Subjekte, Prädikate und Objekte zu identifizieren. Diese haben die Eigenschaft, dass sie von unterschiedlichen Personen und Organisationen unabhängig voneinander entworfen werden können [5]. RDF stellt diese Tripel als Graphen dar, wobei Subjekt und Objekt jeweils durch einen Knoten und das Prädikat durch eine gerichtete Kante vom Subjekt zum Objekt repräsentiert werden (siehe Beispiel).

Ein Beispiel

Die Aussagen „es gibt eine Person die durch <http://www.w3.org/People/EM/contact#me> identifiziert ist, deren Name Eric Miller ist, deren Email-Adresse em@w3.org, und deren Titel Dr. ist“ wird durch den RDF Graphen in Abb.1 repräsentiert.

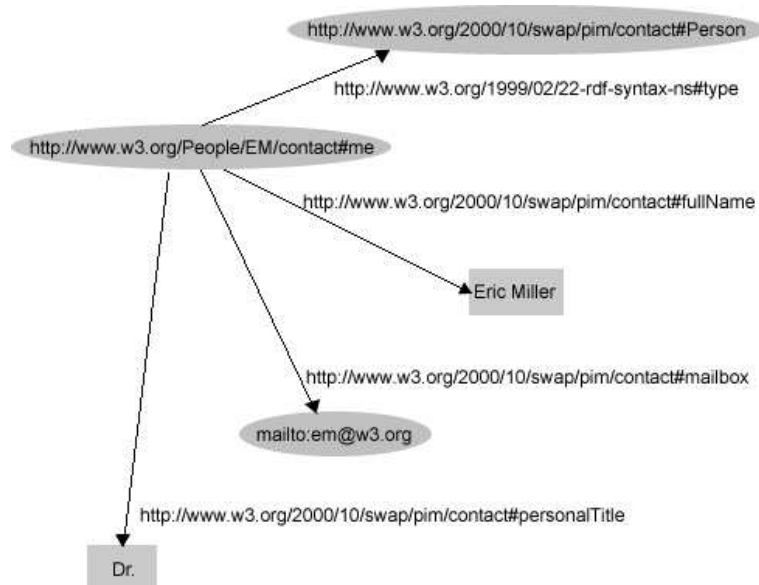


Abbildung 1: Ein Eric Miller beschreibender RDF Graph (aus [5])

2.2 RDF/XML

Um die Graphenmodelle zwischen unterschiedlichen Anwendungen und Plattformen auszutauschen, werden sie in XML-basierter Syntax (RDF/XML) kodiert. Folgender RDF/XML Code beschreibt den Graphen aus obigem Beispiel:

```
<?xml version="1.0">
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:contact="http://www.w3.org/2000/10/swap/pim/contact#">
    <contact:Person rdf:about="http://www.w3.org/People/EM/contact#me">
        <contact:fullName>Eric Miller</contact:fullName>
        <contact:mailbox rdf:resource="mailto:emw3.org"/>
        <contact:personalTitle>Dr.</contact:personalTitle>
    </contact:Person>
</rdf:RDF>
```

Somit zeichnet sich RDF durch drei wesentliche Eigenschaften aus:

Unabhängigkeit : Jeder kann seine eigenen Beschreibungen entwerfen.

Austauschbarkeit : durch die Kodierung in RDF/XML ist die Portabilität im Netz und zwischen Anwendungen gewährleistet.

Skalierbarkeit : die Repräsentation in Tripeln ermöglicht ein leichtes Zugreifen und Verwalten der Daten.

3 RDFCore

Seit RDF 1999 den Status einer W3C Recommendation erhielt [1], sind zahlreiche Application Programming Interfaces (APIs) entwickelt worden um RDF-basierte Anwendungen zu unterstützen. RDFCore wurde in erster Linie entwickelt, um die Benutzung mehrere APIs gleichzeitig zu ermöglichen. Ferner unterstützt es mehrere Anfragesprachen, so dass auch ein Austausch von Anfragen zwischen Systemen mit unterschiedlichen APIs möglich ist. Desweiteren bietet RDFCore ein Multi-User-Environment mit der Möglichkeit Gruppen zu bilden und Lese- sowie Schreibrechte zu vergeben.

3.1 Architektur

Abbildung 2 zeigt die Hauptkomponenten von RDFCore.

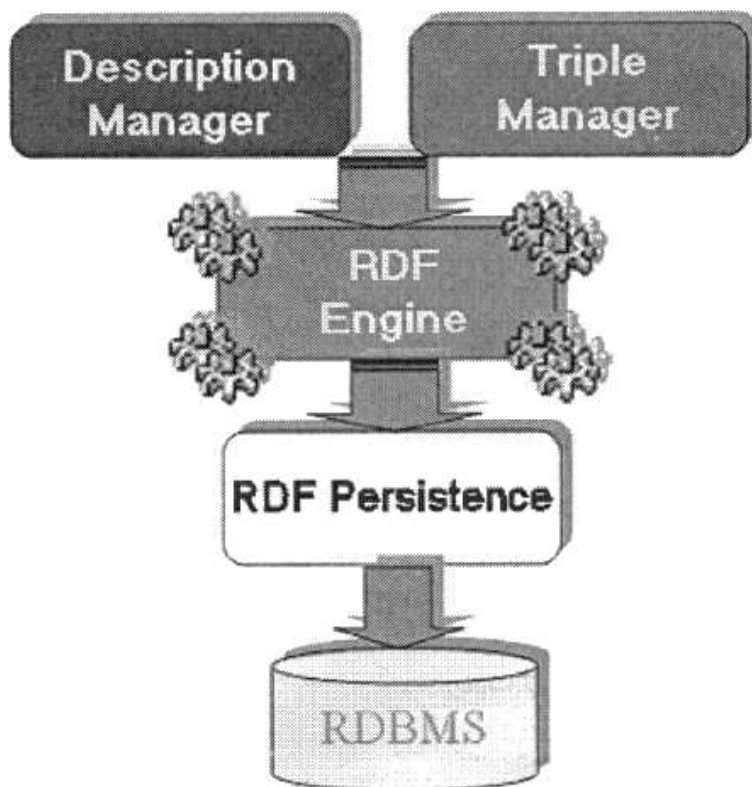


Abbildung 2: Architektur von RDFCore (aus [4])

Description Manager und Tripel Manager

Sie bilden die Benutzungsschnittstelle des Programms. Es wird in Beschreibungen (Mengen von Tripeln, auch Modelle genannt) und Tripel selbst unterschieden um den verschiedenen Zugriffstiefen auf RDF-Ressourcen gerecht zu werden.

Der Description Manager ermöglicht das

- Hinzufügen, Löschen, Zugreifen auf Modelle im eigenen Verzeichnis
- Aktualisieren eines gesamten Modells
- Anfragen an ein oder mehrere Modelle stellen.

während der Tripel Manager das Löschen, Hinzufügen und Ändern von einem oder mehreren Tripeln ermöglicht.

RDF Engine und RDF Persistence

Beim RDF Engine handelt es sich um ein Interface. Er spezifiziert die für die Funktion der Manager notwendigen Operationen. Jeder Aufruf einer Managerfunktion wird in eine Kombination von RDF Engine Operationen übersetzt. Die eigentliche Implementierung dieser ist jedoch der RDF Persistence Ebene überlassen. Durch diese Architektur wird die gleichzeitige Verwendung unterschiedlicher APIs ermöglicht. Mehrere Implementierungen des RDF Engine können nebeneinander existieren.

Ferner kann für jede Benutzeranforderung die optimale RDF Persistence verwendet werden. Es muss lediglich ein RDF Engine implementiert werden, der als Brücke zwischen der Persistence und den Managern fungiert. Bisher wurden zwei verschiedene Engines bezüglich der RDF Modell Speicherung verwirklicht. Einer basierend auf RDF/XML Darstellung, ein weiterer basierend auf Tripel Speicherung in RDBMS (siehe auch Kapitel 3.2).

Enhanced Query Engine

Da es bislang noch keinen Standard bezüglich RDF Anfragesprachen gibt, ist der Austausch von Datenanfragen zwischen Systemen die unterschiedliche APIs verwenden schwierig. Das Enhanced Query Engine Modul ist eine Subkomponente und bietet die Möglichkeit verschiedene Anfragesprachen zu verwenden. Es ist so aufgebaut, dass das Hinzufügen einer neuen Anfragesprache mit möglichst geringem Aufwand verbunden ist.

3.2 Performanz Test

Da es sich bei RDFCore im wesentlichen um ein Speicherungssystem für Wissen handelt, ist vor allem seine Skalierbarkeit hinsichtlich der zu verwaltenden Datenmenge von Interesse. Es wurden empirische Tests durchgeführt um das Laufzeitverhalten bei steigender Datenmenge zu ermitteln. Wie bereits oben erwähnt wurden zwei verschiedene Arten der Datendarstellung und Speicherung entwickelt:

- basierend auf Binärspeicherung der RDF/XML Darstellung in PDOM
- basierend RDBMS Speicherung von RDF-Ressourcen

Da beide Implementierungen Mechanismen besitzen um Datenredundanz zu ihrem Vorteil zu nutzen, wurden zwei stetig wachsende Testmengen entworfen.

A : hohe Redundanz, Vergrößerung der Datenmenge durch Wiederholung des RDF Modells

B : keine Redundanz, Vergrößerung der Datenmenge durch Hinzufügen neuer Tripel, d.h. ohne Subjekt, Objekt oder Prädikat zu wiederholen.

Untersucht wurde das Laufzeitverhalten beider Implementierungen beim

- Einlesen der Daten (Speicherzeit)
- Hinzufügen eines Tripels in einen großen Datensatz
- Zugriff auf ein RDF Modell
- Abfrage jedes Tripels eines Modells

3.2.1 Ergebnisse

Im Folgenden sind Dateien mit Namen *NNNx_rdf* redundante Beschreibungen, wobei *NNN* die Anzahl der Wiederholungen eines Tripels ist. Dateien mit Namen *Output-NNNNN_rdf* sind Beschreibungen ohne Redundanz, wobei *NNNNN* die Anzahl der Tripel ist. In allen folgenden Abbildungen ist die Skalierung der Y-Achse logarithmisch, die Einheit der Laufzeit ist Millisekunde (msec). Tabelle 1 zeigt das Laufzeitverhalten für das Einlesen der Testmenge A. Abbildungen 3 und 4 zeigen das Laufzeitverhalten im Vergleich zu einer theoretischen, linearen Funktion bezüglich der Datenmenge für die PDOM Implementierung. Abbildungen 5 und 6 zeigen das gleiche für die RDBMS Implementierung.

File	File size (Kbytes)	PDOM Persistence			JENA Persistence		
		Elapsed time (milliseconds)	Theoretical elapsed time	PDOM file size	Reading time	Storing time	Theoretical storing time
2x.rdf	173	3886	4000	-	203	9777	10000
3x.rdf	259	4016	6000	-	250	14772	15000
4x.rdf	342	4226	8000	-	313	19779	20000
5x.rdf	432	4446	10000	-	453	24767	25000
6x.rdf	518	4827	12000	-	485	29787	30000
7x.rdf	605	4547	14000	-	563	34787	35000
8x.rdf	691	5178	16000	-	640	39766	40000
9x.rdf	777	4757	18000	-	734	44822	45000
10x.rdf	864	5417	20000	-	875	49822	50000
11x.rdf	950	5117	22000	-	953	54762	55000
12x.rdf	1036	5168	24000	-	1125	59783	60000
13x.rdf	1122	5007	26000	-	1062	64747	65000
14x.rdf	1209	5488	28000	-	1250	69827	70000
15x.rdf	1295	5427	30000	-	1234	74727	75000
16x.rdf	1381	5948	32000	-	1312	79837	80000
17x.rdf	1468	5728	34000	-	1422	84737	85000
18x.rdf	1554	5808	36000	-	1547	89737	90000
19x.rdf	1640	5879	38000	-	1547	94687	95000
20x.rdf	1727	5929	40000	-	1656	99682	100000
30x.rdf	2590	7411	60000	261	2390	149573	150000
40x.rdf	3453	9043	80000	314	3250	199394	200000
50x.rdf	4316	10104	100000	370	4015	249230	250000
60x.rdf	5179	10905	120000	427	4781	299171	300000
70x.rdf	6042	11636	140000	482	5578	348987	350000
80x.rdf	6905	13930	160000	532	6453	398823	400000
90x.rdf	7768	13450	180000	584	7203	448658	450000
100x.rdf	8631	14130	200000	638	9437	498494	500000
110x.rdf	9494	15292	220000	691	9406	548403	550000
120x.rdf	10357	15793	240000	744	10343	598239	600000
130x.rdf	11220	18807	260000	797	11032	648011	650000
140x.rdf	12083	13450	280000	848	11688	697917	700000
150x.rdf	12946	22272	300000	900	12594	747847	750000
160x.rdf	13809	21802	320000	952	13469	797643	800000
170x.rdf	14672	23384	340000	1003	14469	847636	850000
180x.rdf	15535	24105	360000	1055	15469	897452	900000
190x.rdf	16398	25317	380000	1106	17438	947425	950000
200x.rdf	17261	26201	400000	1157	16891	997201	1000000

Tabelle 1: Laufzeitverhalten beim Einlesen von Daten mit hoher Redundanz (aus [4])

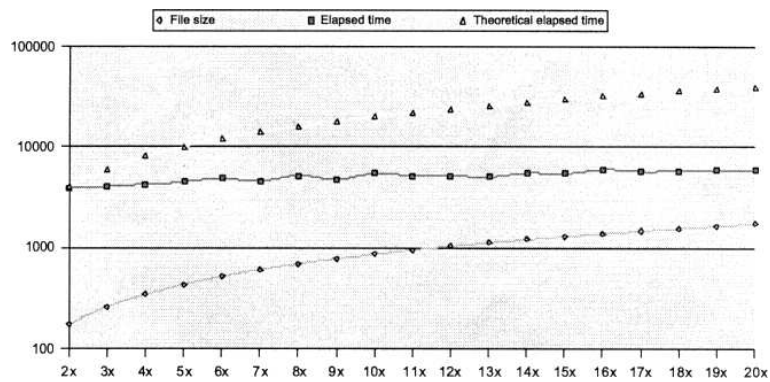


Abbildung 3: Graphische Darstellung des Laufzeitverhalten der PDOM Implementierung beim Einlesen von Daten mit hoher Redundanz (aus[4])

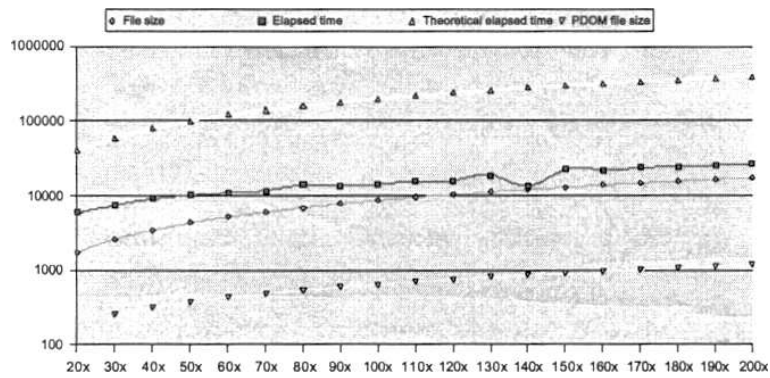


Abbildung 4: Graphische Darstellung des Laufzeitverhalten der PDOM Implementierung beim Einlesen von Daten mit hoher Redundanz (aus [4])

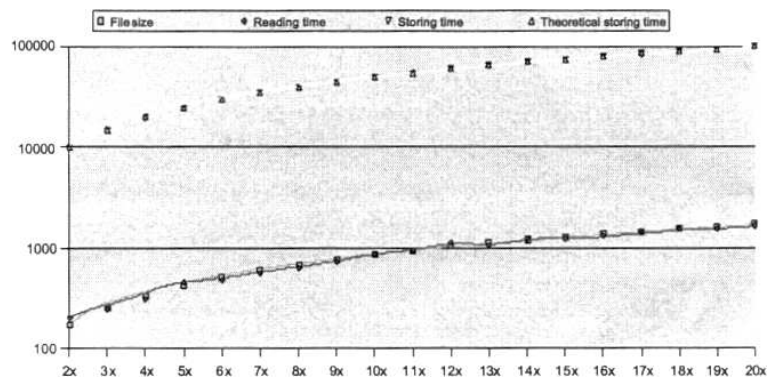


Abbildung 5: Graphische Darstellung des Laufzeitverhalten der RDBMS Implementierung beim Einlesen von Daten mit hoher Redundanz (aus [4])

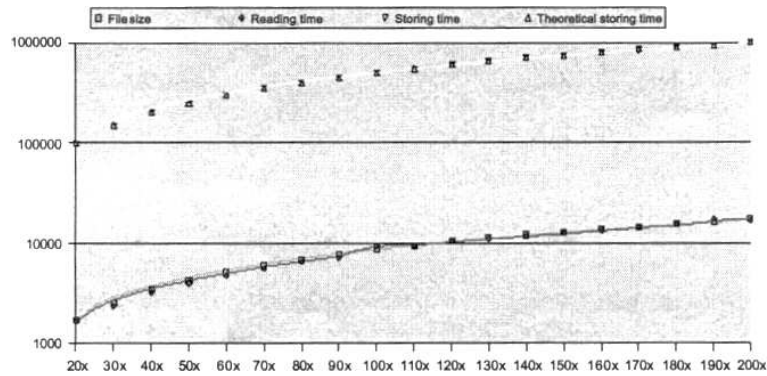


Abbildung 6: Graphische Darstellung des Laufzeitverhalten der RDBMS Implementierung beim Einlesen von Daten mit hoher Redundanz (aus [4])

Tabelle 2 zeigt das Laufzeitverhalten für das Einlesen der Testmenge B, Abbildungen 7 und 8 zeigen die graphische Darstellung der Daten.

File	File size	PDOM Persistence				JENA Persistence		
		PDOM file size	Reading time	Storing time	Theoretical storing time	Reading time	Storing time	Theoretical storing time
Output10000	1480	1210	6990	15382	15000	2219	83612	80000
Output20000	2990	2470	10404	26689	30000	3140	167201	160000
Output30000	4490	3700	15682	36823	45000	4797	250750	240000
Output40000	6000	4970	19999	48139	60000	6125	334200	320000
Output50000	7510	6210	26178	59776	75000	7828	418035	400000
Output60000	9000	7450	29893	76700	90000	9422	501715	480000
Output70000	10500	8700	34089	99152	105000	13281	585204	560000
Output80000	12000	9920	38305	145219	120000	15328	667835	640000
Output90000	13500	11100	45174	208650	135000	16531	752483	720000
Output100000	15000	12400	49812	308293	150000	18438	836212	800000

Tabelle 2: Laufzeitverhalten beim Einlesen von Daten ohne Redundanz (aus [4])

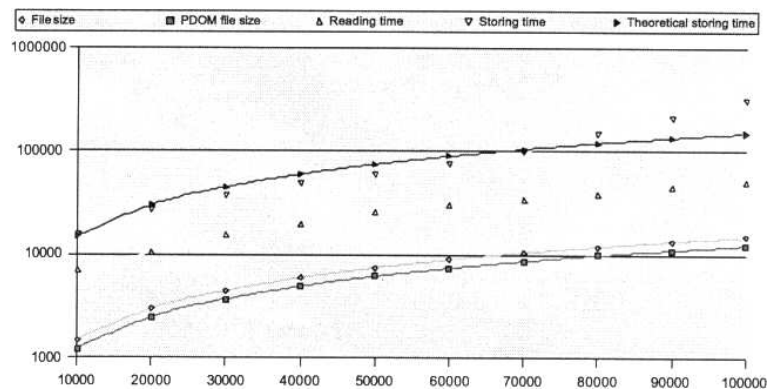


Abbildung 7: Graphische Darstellung des Laufzeitverhalten der PDOM Implementierung beim Einlesen von Daten ohne Redundanz (aus [4])

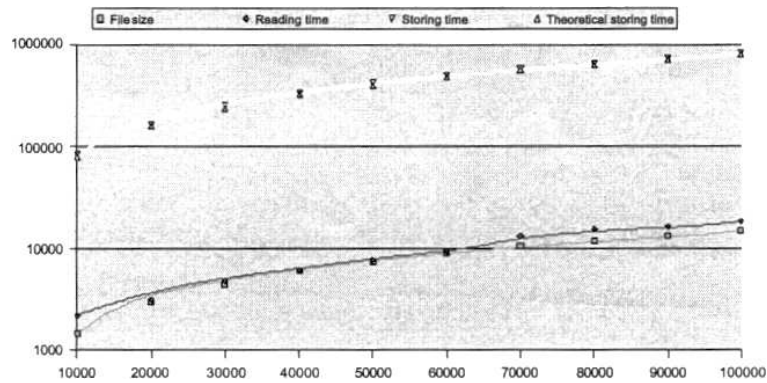


Abbildung 8: Graphische Darstellung des Laufzeitverhalten der RDBMS Implementierung beim Einlesen von Daten ohne Redundanz (aus [4])

Tabelle 3 sowie Abbildungen 9 und 10 zeigen das Laufzeitverhalten für das Einfügen eines Tripels in große Datensätze, während in Tabelle 4 sowie Abbildung 11 und 12 die Zeit aufgeführt ist, die RDFCore benötigt um auf ein RDF Modell zuzugreifen und es für die Bearbeitung durch den Benutzer bereitzustellen.

PDOM Persistence			JENA Persistence		
File	Elapsed time	Theoretical elapsed time	File	File size	Elapsed time
160x.rdf	9333	9333	Output10000	1480	358
170x.rdf	9564	9916	Output20000	2990	12
180x.rdf	10826	10500	Output30000	4490	25
190x.rdf	10756	11082	Output40000	6000	70
-	-	-	Output50000	7510	36
-	-	-	Output60000	9000	10
-	-	-	Output70000	10500	17
-	-	-	Output80000	12000	21
-	-	-	Output90000	13500	20
-	-	-	Output100000	15000	20

Tabelle 3: Laufzeitverhalten für das Einfügen eines Tripels in große Datensätze (aus [4])

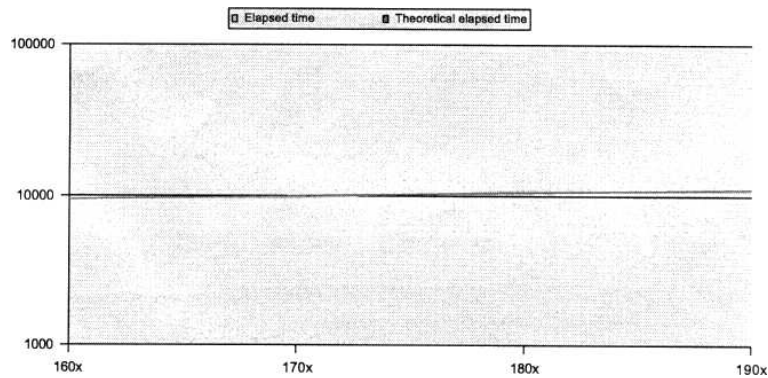


Abbildung 9: Graphische Darstellung des Laufzeitverhalten der PDOM Implementierung für das Einfügen eines Tripels in große Datensätze (aus [4])

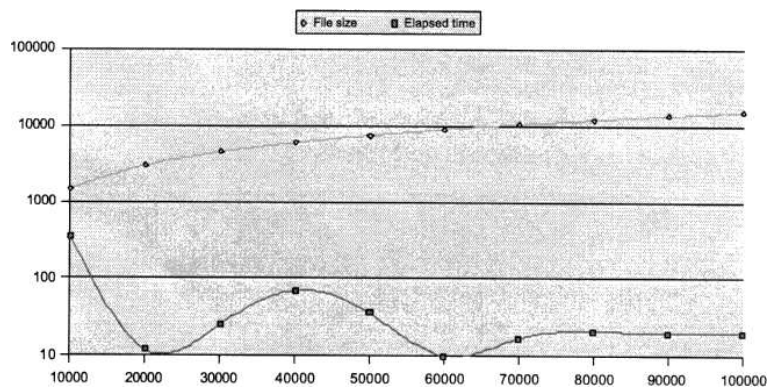


Abbildung 10: Graphische Darstellung des Laufzeitverhalten der RDBMS Implementierung für das Einfügen eines Tripels in große Datensätze (aus [4])

Resource	PDOM Persistence		JENA Persistence
	Elapsed time	Theoretical elapsed time	Elapsed time
Output10000	13570	13000	484
Output20000	23804	26000	5
Output30000	34420	39000	15
Output40000	43573	52000	63
Output50000	59285	65000	31
Output60000	-	-	5
Output70000	-	-	7
Output80000	-	-	15
Output90000	-	-	16
Output100000	-	-	16

Tabelle 4: Laufzeitverhalten für das Bereitstellen eines RDF Modells zur Bearbeitung durch den Benutzer (aus [4])

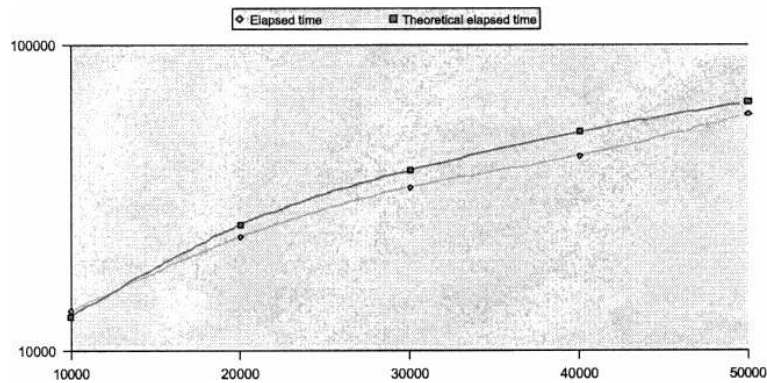


Abbildung 11: Graphische Darstellung des Laufzeitverhalten der PDOM Implementierung für das Bereitstellen eines RDF Modells zur Bearbeitung durch den Benutzer (aus [4])

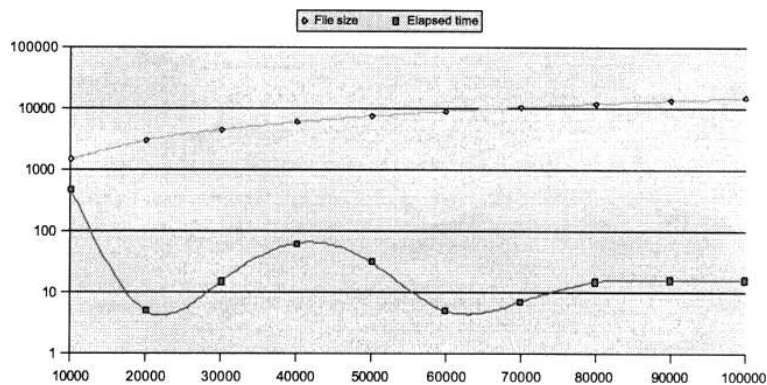


Abbildung 12: Graphische Darstellung des Laufzeitverhalten der RDBMS Implementierung für das Bereitstellen eines RDF Modells zur Bearbeitung durch den Benutzer (aus [4])

Tabelle 5, Abbildung 13 und 14 zeigen das Laufzeitverhalten von RDFCore bezüglich der Abfrage aller Tripel eines Modells. Bei dieser Abfrage wird das Enhanced Query Modul verwendet. Die an das System gestellte Anfrage lautete alle Tripel zurückzugeben, die einen Wert für Subjekt, Prädikat und Objekt besitzen. In der verwendeten Anfragesprache RDQL sieht das so

```
SELECT ?s, ?p, ?o WHERE (?s,?p,?o)
```

aus.

Resource	Triple number	PDOM Persistence	JENA Persistence
		Elapsed time	Elapsed time
Output10000_rdf	10000	10505	453
Output20000_rdf	20000	15502	31
Output30000_rdf	30000	24075	16
Output40000_rdf	40000	32497	15
Output50000_rdf	50000	49361	16

Tabelle 5: Laufzeitverhalten für die Abfrage jedes Tripels eines Modells (aus [4])

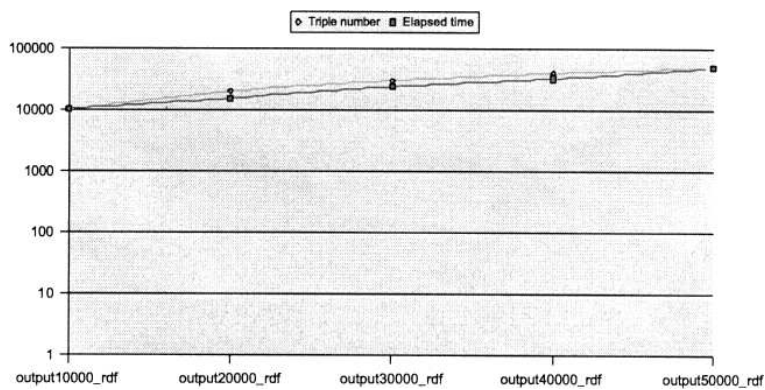


Abbildung 13: Graphische Darstellung des Laufzeitverhalten der PDOM Implementierung für die Abfrage jedes Tripels eines Modells (aus [4])

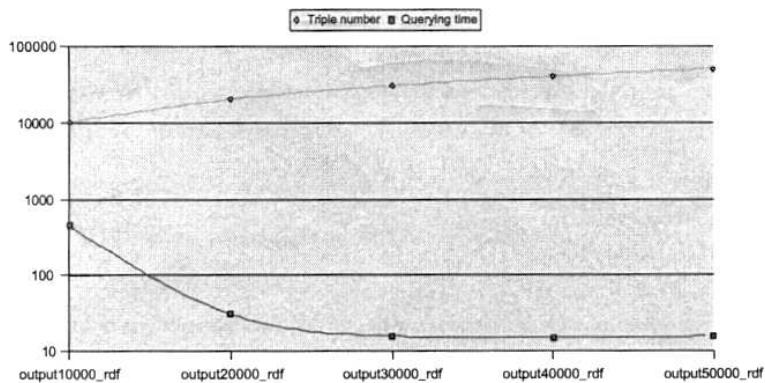


Abbildung 14: Graphische Darstellung des Laufzeitverhalten der RDBMS Implementierung für die Abfrage jedes Tripels eines Modells (aus [4])

3.2.2 Analyse

Das Laufzeitverhalten des gesamten Systems ist für beide Implementierungen linear zur Datenmenge. Lediglich das Laufzeitverhalten der PDOM Implementierung könnte beim Einlesen sehr großer Datenmengen ohne Redundanz nicht linear sein (vergl. Tab.2 und

Abb.7). Hier wäre es interessant gewesen noch größere Datenmengen zu testen, um ein exponentielles Ansteigen der Speicherzeit sicher auszuschließen. Da der Hersteller des verwendeten PDOM sein Produkt für Datenmengen im Gigabyte-Bereich anpreist ([6]), liegt evtl. ein Fehler im Testverfahren vor.

Das Laden eines neuen Modells dauert bei der RDBMS Implementierung aufgrund der Komplexität der internen Datenbankstruktur wesentlich länger als bei der PDOM Implementierung. Andererseits zeigt diese Implementierung beim Zugriffs- und Anfragetest fast konstantes Laufzeitverhalten, während die PDOM Implementierung hier in Linearzeit arbeitet. Dies ist auf die unterschiedliche interne Datenverwaltung zurückzuführen. Während PDOM die Daten in den Hauptspeicher laden muss, kann die RDBMS Implementierung auf ihre Datenbankstruktur zurückgreifen, was offensichtlich schneller ist. Ungeklärt bleibt leider warum die Laufzeit der RDBMS Implementierung beim Hinzufügen eines Tripels (Tab.3, Abb.10), Zugreifen auf ein Modell (Tab.4, Abb.12) und bei der Abfrage aller Tripel eines Modells (Tab.5, Abb.14) zu Beginn einer Testserie geringerer Datenmenge höher ist, als später, bei größerer Datenmenge. Hier scheint ein Fehler im Testverfahren vorzuliegen, evtl. werden zu Beginn einer Testserie noch zusätzliche Operationen auf der Datenbank ausgeführt (Initialisierung, ect.).

Unklar ist auch, ob beim Anfragetest (Tab.5) immer das gleiche Modell abgefragt wurde, und somit immer gleich viele Tripel aus einer wachsenden Datenmenge zurückgegeben werden mussten, oder ob das Modell jeweils die gesamte Datenmenge beinhaltet. Wäre letzteres der Fall ist zu klären, warum bei der RDBMS Implementierung das Zurückgeben von 50.000 Tripeln genau so lange dauert wie das Zurückgeben von 30.000 Tripel, zumal das Lesen einer Datei mit 50.000 Tripeln bereits deutlich länger dauert, als das Lesen einer Datei mit 30.000 Tripeln (vergl. Tab.2).

3.3 Anwendung

Ein Anwendungsbeispiel von RDFCore ist das EU Forschungsprojekt COLLATE [3]. In diesem Projekt geht es darum ein System für die Verwaltung von Filmproduktionen aus Österreich, Deutschland und den Tschechischen Republiken in den 30er Jahren zu Forschungszwecken, zu entwickeln. Drei Archive müssen elektronisch zugänglich gemacht werden, und Wissenschaftler müssen die Möglichkeit haben Inhalte zu Katalogisieren, Kommentieren, ect. Angesichts der großen Datenmenge ist ein effizientes Verwaltungssystem dringed notwendig. RDFCore mit einer RDF Persistence Implementierung die auf Darstellung der RDF Tripel in einem RDBMS basiert ermöglicht ein schnelles Arbeiten, da Anwendungen nicht das gesamte RDF Modell laden müssen, um mit wenigen Tripeln daraus zu arbeiten. Ferner bietet es mit dem Enhanced Query Engine die Möglichkeit der API übergreifenden Anfrage.

4 Zusammenfassung und Ausblick

RDFCore bietet interessante Möglichkeiten zum Verwalten von RDF Modellen. Gerade die Option des Multi-User-Environments ist im Hinblick auf das World Wide Web von

großem Interesse, da sie mehreren Autoren einer Seite das Erstellen eines zugehörigen RDF Modells sehr erleichtern kann.

Die Anwendung im Forschungsprojekt COLLATE zeigt, dass RDF und darauf aufbauende Technologien auch ausserhalb des World Wide Web sinnvolle Lösungen bieten können. Für die Zukunft ist geplant höhere Programmiersprachen wie DAML+OIL in RDFCore einzubinden und eine Standard RDF Anfragesprache zu unterstützen sobald sie vom W3C herausgegeben wird.

5 Quellen

- [1] BRAY, T.: What is RDF? <http://www.xml.com/pub/a/2001/01/24/rdf.html>, 2001
- [2] CHAMPIN, P.-A.: RDF Tutorial. <http://www710.univ-lyon1.fr/~champin/rdf-tutorial/>, 2001
- [3] COLLATE - COLLATE- Collaboratory for Annotation, Indexing and Retrieval of Digitized Historical Archive Material <http://www.collate.de/>
- [4] ESPOSITO, F. et al.: RDFCore : A component for effective management of RDF Models. ???????
- [5] MANOLA, F. & MILLER, E. (editors): RDF Primer. <http://www.w3.org/TR/rdf-primer/>, 2003
- [6] http://www.infonyte.com/en/prod_pdom.html