FlyingDoc: Eine Architektur für verteilte, nutzerfreundliche und personalisierte Informationssysteme

Andre Zeitz, Ilvio Bruder
Universität Rostock, Institut für Informatik
Lehrstuhl Datenbank- und Informationssysteme

Email: {zeitz, ilr}@informatik.uni-rostock.de

Zusammenfassung

Aktuelle Informationssysteme realisieren viele Features und gute Retrieval-Charakteristiken. Sie sind oft sehr komplex und ohne spezielles Wissen nicht einfach anzuwenden. Nutzerfreundliche Informationssysteme bieten meist nur eingeschränkte Funktionalität. FlyingDoc bietet Konzepte und eine Architektur für ein nutzerfreundliches, verteiltes und personalisiertes Informationssystem. Jeder Nutzer soll State-of-the-Art Retrieval und Zugriffsmethoden ohne Spezialwissen nutzen können. Ein getestetes Anwendungsszenario der FlyingDoc-Architektur ist ein persönlicher Konferenzplaner. Dieser wurde z.B. auf der VLDB 2003 in Berlin getestet.

1 Einleitung

Der Bedarf zur Verwaltung von Informationen ist steigend, insbesondere zur verteilten, personalisierten und vor allem nutzerfreundlichen Verwaltung von Dokumenten. Digitale Bibliotheken als aktuelle Vertreter der Dokumenten-Management-Systeme bieten Unterstützung für den kompletten Lebenszyklus von Dokumenten. Um diese Systeme derart zu erweitern, dass sie verteilt nutzbar, personalisierbar und nutzerfreundlich werden, stellen wir hier Techniken vor, die einen einfachen Import neuer Dokumentenkollektionen ermöglichen, das Anlegen neuer Datenmodelle unterstützen und aufgrund von Personalisierung und Replikation mobil einsetzbar sind.

Als eine konkrete Implementierung dieser Techniken soll hier ein System vorgestellt werden, das als Konferenz-Management-Tool auf der VLDB 2003 in Berlin eingesetzt wurde (VLDB 2003 eClient[1]). Außerdem soll hier auf Tools (FlyingDoc) eingegangen werden, die auf die Verwaltung persönlicher, lokaler Dokumentenbestände abzielen und unabhängig vom VLDB eClient entwickelt wurden.

In diesem Beitrag stellen wir zunächst eine Architektur und ein Anwendungsszenario vor. Nach einer Erörterung verwandter Arbeiten beschreiben wir selbst entwickelte Techniken. In Abschnitt 4 stellen wir eine Implementierung vor, bevor der Artikel mit einer Zusammenfassung und einem Ausblick schließt.

2 Architektur

Ausgehend von zwei Szenarien, einem Digitalen Bibliothekssystem und einem Konferenzinformationssystem, ist der folgende Ansatz konzipiert. In beiden Szenarien soll es um die Dokumentverwaltung auf Nutzerseite gehen bzw. um ein Client-basiertes Informationssystem mit Support durch einen Informationsserver.

Ähnliche Ansätze finden sich vor allem im Bereich der Dokumenten- und Content-Management-Systeme und auch im Anwendungsbereich der Digitalen Bibliotheken. Insbesondere sind Modelle, Replikationstechniken und Personalisierung von Interesse. Schon in frühen Digitalen Bibliotheksarchitekturen, wie DIENST[2] [3] wurden Anforderungen für Kollektionen diskutiert. Die Service-Architektur für Kollektionen (die sogenannte FEDORA-Objektmodell und Repository-Architektur) beinhaltet eine Unterstützung für das Importieren heterogener Dokumente und die Aggregation gemischter Daten in komplexe Objekte.

[1] http://wwwdb.informatik.uni-rostock.de/vldb2003/eClient/
[2] http://www.cs.cornell.edu/cdlrg/dienst/DienstOverview.htm

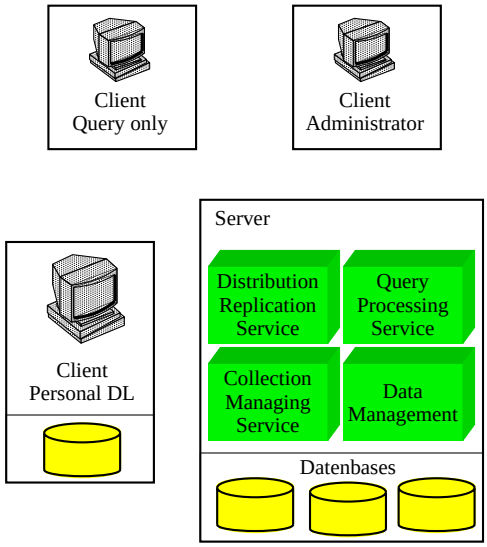

Abbildung 1: Architekturüberblick

Die Client-Server-Architektur in Abb. 1 veranschaulicht einen Informationsserver mit verschiedenen Services und drei verschiedenen Arten von Clients. Die interessantesten sind Clients, die ihr eigenes Informationssystem verwenden. Diese beinhalten eine persönliche Dokumentverwaltung, um eigene, selbst importierte oder per Download erhaltene Dokumente zu verwalten. Zusätzlich können Werkzeuge und Schnittstellen für die Anbindung und integrierte Verwendung von Services eines Servers benutzt werden.

Die Komponenten des Servers bestehen aus einem Verteilungs- und Replikationsservice, einem Service zur Integration von Kollektionen, einer Datenverwaltungseinheit und einer Anfrageverarbeitung. Die Verteilungs- und Replikationskomponente besteht aus einem Verbindungsmanagement für die Clients (mit eigenem Informationssystem) und Verteilungsaspekten für die Kollektionen, deren Indizes, Standardanfragen und Standardrepräsentationen. Der Service zur Integration externer Kollektionen kann neue Kollektionen einbinden oder bestehende aktualisieren. Dabei wird beim Dokumentimport eine Dokument- und Schema-Analyse verwendet. Die Datenverwaltungseinheit verwaltet Metadaten, Objekt- und Strukturdaten sowie die Indexdaten. Die Anfrageverarbeitung bietet neben der eigentlichen Anfrageverarbeitung auch die automatische Generierung von Standardanfragen und Standard-Datenrepräsentationen.

3 Techniken zum personalisierten, verteilten und nutzerfreundlichen Dokumenten-Management

In diesem Abschnitt werden einige Aspekte des FlyingDoc-Systems diskutiert. Diese umfassen insbesondere die Datenmodellierung, den Import von Daten bzw. Dokumenten. Auf Mobilität, Replikation und Personalisierung soll außerdem eingegangen werden.

3.1 Modellierungs- und Importaspekte

Eines der Ziele unseres Systems besteht darin, ein Datenmodell zu finden, mit dem nahezu alle Daten beschrieben werden können. Dieses Modell sollte so einfach wie möglich aufgebaut, leicht in einer Datenbank zu speichern sein und eine einfache Abbildung auf andere Datenmodelle ermöglichen.

Viele Dokumentenkollektionen werden heutzutage in Digitalen Bibliotheken wie etwa Greenstone [6] gespeichert, das aus dem *New Zealand Digital Library Project*[3] hervorgegangen ist. Das

[3] http://www.nzdl.org

interne Datenmodell von Greenstone besteht aus relativ einfachen Definitionen, die sowohl Metadaten als auch strukturelle Beschreibungen umfassen können. Solche Beschreibungen werden mit Hilfe von XML durchgeführt, wobei die dazugehörige DTD lediglich vier Elemente umfasst. Um Dokumente auf das Greenstone-Modell abzubilden müssen innerhalb der Dokumente (XML-Dokumente) spezielle Mapping-Elemente definiert werden.

Unser Datenmodell, dass in [7] vorgestellt wurde, stellt einen Kompromiss zwischen einem rein generischen und einem spezialisierten, starren Ansatz dar. Ein rein generisches Modell würde aus relativ wenigen Datentypen oder sogar nur einem einzigen Datentyp bestehen, der alle möglichen Datentypen repräsentieren kann. Diese Art der Modellierung führt jedoch zu großen, entarteten Daten, da die inneren Strukturen sehr umfangreich und Suchanfragen auf diesen Strukturen sehr kompliziert werden können. Ein anderer Ansatz besteht darin, ein spezielles Modell zu bilden, das aus einer großen Anzahl von Datentypen besteht. Ein solches Modell ist zwar einfach benutzbar und anfragbar, aber es muss immer dann angepasst werden, wenn neue Datenstrukturen damit modelliert werden sollen. Ein weiterer Ansatz ist in [4] beschrieben. Dort werden Daten hybrid modelliert, wobei objektorientierte und semistrukturierte Elemente innerhalb eines Modells vorkommen können.

Unser Datenmodell besteht aus zwei Teilen: Metadaten und Objektdaten. Metadaten sind dabei hierarchisch als Baum organisiert. Die Knoten eines solchen Baumes besitzen jeweils einen impliziten Typ, einem Namen und einem optionalen Wert. Der Typ des Wertes hängt vom Knotentyp ab. Eine Erweiterung dieses Modells beinhaltet einen Mechanismus, um Abhängigkeiten zwischen verschiedenen Knoten nachzubilden, was beispielsweise zur Wahrung der referentiellen Integrität innerhalb zu importierender Daten wichtig sein kann. Objektdaten werden ebenfalls als Baum strukturiert. Hier wird zwischen zwei Knotenarten unterschieden: inneren Knoten und Blattknoten. Innere Knoten enthalten einen Namen, eine optionale Menge von Kindknoten, die wiederum innere Knoten sind, und eine Menge von Attributen. Blattknoten werden genutzt, um Teile eines Dokumentes wie Abschnitte, Kapitel oder Abbildungen zu modellieren. Metadaten können an Dokumententeile gehängt werden, wodurch jeder Dokumentenanteil eigene Metadaten haben kann. Zusätzlich können Metadaten auch entlang der Baumstruktur vererbt werden.

Eine Kollektion wird mit Hilfe des Import-Dienstes in das System geladen. Sie wird durch eine Menge von Schemata bestimmt. Der Importprozess integriert neben dem Laden von Kollektionen auch Analysetechniken. Ein Hauptanliegen für das System besteht darin, eine nutzerfreundliche und einfache, aber effektiv handhabbare Verwaltung von Kollektionen zu ermöglichen. Dieser Aspekt wurde u.a. in [5] kontrovers diskutiert. Wir bieten deshalb für die Deklaration eines Imports sehr einfache Regeln an, z.B. ist das Festlegen einiger wichtiger Elementnamen oft schon ausreichend. Bei der Analyse versuchen wir durch Lexika bestimmte Metadaten, wie Dublin Core, zu erkennen. Insbesondere wird die Analyse verwendet, um für die Verwaltung der Daten wichtige Informationen zu bestimmen, wie Struktur- und Layoutdaten. Diese Analyse-Informationen werden für die Kollektionsverwaltung, für die Modellierung und Generierung von Standardanfragen und Präsentationstemplates verwendet. Möglich ist es auch, mittels der Dokumentstruktur Dokumentteile zu bestimmen, die beim Suchen in großen Dokumenten zum Fokussieren helfen.

Bei einem Import wird zuerst mittels Schema-Analyse das Schema extrahiert. Danach wird anhand der durch den Nutzer definierten Abbildungsregeln die Abbildung des Modells der zu importierenden Daten auf unser Modell vorgenommen. Um aus den Daten, die unserem Modell vorliegen wieder die Daten zu generieren, die ursprünglich importiert wurden, werden während des Imports auch bestimmte Informationen wie das Layout extrahiert und gespeichert.

3.2 Mobilität und Replikation

Mobilität ist ein essentielles Konzept bei der Erstellung von heutigen Informationssystemen wie beispielsweise Digitalen Bibliotheken. Mobilität umfasst etwa die Möglichkeit von Nutzern, an

verschiedenen Orten mit verschiedenen Geräten mit denselben Daten arbeiten können. Mobilität bedeutet auch, dass Nutzer verschiedene Rollen einnehmen können, wie die eines Autors oder eines Lesers. Im allgemeinen wird Mobilität durch eine dynamische Nutzerumgebung charakterisiert, dem *mobilen Kontext*. Ein solcher mobiler Kontext umfasst verschiedene Objektkontexte (Metaschemata, Abhängigkeiten, Prioritäten, Zeitstempel usw.), eine statische und dynamische Nutzerumgebung (erreichbare Digitale Bibliotheken und Datenbanken, Informationen über Ort, Zeit oder Historie) und Ressourceninformationen über das vorhandene Netzwerk, lokale und entfernte Geräte (Bandbreite, Speicher, Netzwerkverbindung, Darstellungsmöglichkeiten u.a.).

Um in diesem Zusammenhang Digitale Bibliotheken aufbauen zu können, sind Techniken notwendig, die Transparenz und kontext-sensitive Adaptionen zum Speichern, Verwalten und Anfragen von Daten bieten. Adaption kann durch Modifikation von Anfragen oder der Anfrageergebnisse erreicht werden. Eine Datenreduktion lässt sich bezüglich der Datenmenge oder auch der Datenqualität umsetzen.

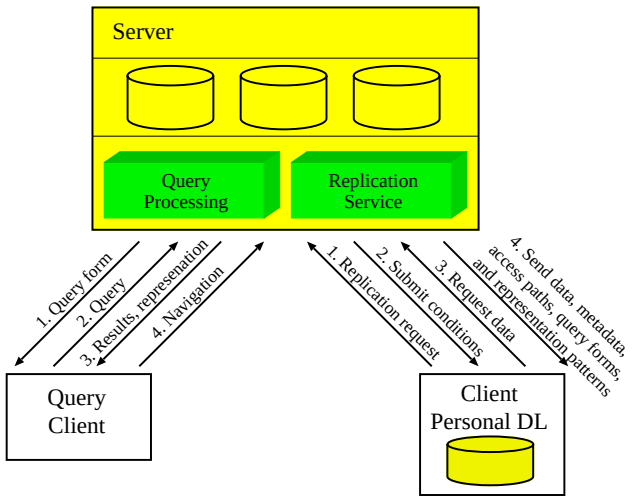

Abbildung 2: Anfrage- and Replikationsschritte

Replikation ist eine weitere Technik, um fehlende Funktionalitäten in mobilen Umgebungen zu überwinden. Durch Replikation wird der Kommunikationsaufwand verringert und gleichzeitig Verfügbarkeit und Lokalität der Daten während des Retrieval-Prozesses erhöht. In einer typischen Digitalen Bibliothek werden drei Arten von Daten repliziert: Metadaten, Indexinformationen (Zugriffspfade) und die Dokumente selbst. Außerdem werden noch Systemfunktionalitäten und die Anfrageverarbeitung (Repräsentationsmuster, Schemata, Anfrageformulare) repliziert bzw. verteilt (siehe Abb. 2).

Unsere Implementierung repliziert alle benötigten Metadaten und Indexinformationen auf den lokalen Client. Welche Daten jeweils benötigt werden, wird dabei vom Nutzerkontext abgeleitet, z.B. von den Aufgaben, die ein Nutzer zu erfüllen hat, den Orten, an denen er sich aufhält oder den technischen Fähigkeiten seines momentan benutzten Gerätes. Die Dokumente, auf die der Nutzer zugreift werden lokale repliziert. Zugriffe darauf können so in Zukunft schneller durchgeführt werden. Diese Voraussage basiert auf denselben Kontextinformationen, die zur Vorhersage der in Zukunft benötigten Meta- und Indexdaten angewendet wird. Indexinformationen und Dokumente können hinsichtlich der Ressourcenbeschränkungen fragmentiert werden. Relevante Fragmente werden dazu genauso bestimmt wie die Ranking-Werte von Retrieval-Ergebnissen.

3.3 Personalisierungsaspekte

Um nach Dokumenten zu suchen und darauf zuzugreifen, müssen Nutzer meist selbst eine Suche innerhalb einer Dokumentenkollektion durchführen. D.h. sie müssen eine Anfrage formulieren

und sich aus der Ergebnismenge für sie relevante Dokumente heraussuchen. Um diesen Vorgang zu vereinfachen unterstützt unser System das Konzept der *Relevanz* bezüglich *Ort* und *Zeit*.

Möchte ein Nutzer beispielsweise auf einer Konferenz alle für ihn relevanten Dokumente der Session einsehen, in der er sich momentan befindet, muss zunächst herausgefunden werden, wo sich der Nutzer gerade aufhält. Leider ist dafür spezielle Hardware notwendig, wie in [1, 2] beschrieben. Um dieses Problem zu umgehen, können Nutzer eine persönliche Route durch die Konferenz festlegen, d.h. welche Vorträge sie sich anhören möchten. Eine Anfrage nach allen relevanten Dokumenten beantwortet das System dann mit Hilfe der aktuellen Zeit, der persönlichen Route und dem Konferenzplan. Der Nachteil dieser Methode ist, dass sie nicht funktioniert, wenn ein Nutzer beispielsweise in einer Session teilnimmt, die er nicht in seine Route eingetragen hat. Andererseits wird das Konferenzprogramm immer aktuell gehalten, so dass Nutzer auch dann die richtige Dokumente erhalten, wenn eine Session verschoben wird.

Da immer nur die Dokumente auf das mobile Gerät repliziert werden, die der Nutzer anfragt, werden Speicherplatz und Bandbreite und somit insbesondere mobile Geräte unterstützt.

4 Implementierung

Einige der hier vorgestellten Konzepte wurden in Form eines Konferenzplaners (VLDB eClient) als Teil von FlyingProc umgesetzt und auf der VLDB 2003 in Berlin eingesetzt. Mit Hilfe dieses Planers können Nutzer wie oben beschrieben nach relevanten Dokumenten (Paper, Vortragsfolien) suchen. Dazu müssen sie interaktiv festlegen, an welchen Sessions sie teilnehmen möchten (siehe Abschnitt 3.3). Eine Volltextsuche über den Artikel und den Vortragsfolien wird auch durch den Planer unterstützt.

Zur Unterstützung mobiler Geräte werden Dokumente, für die sich Nutzer interessieren, lokal repliziert, so dass auch im Offline-Betrieb ein Zugriff darauf möglich ist. Diese replizierten Dokumente werden immer aktuell gehalten, d.h. wenn das mobile Geräte online ist und sich auf dem Server Dokumente geändert haben, dann werden diese lokal aktualisiert.

5 Zusammenfassung und Ausblick

In diesem Beitrag wurden Konzepte und eine Architektur für nutzerfreundliche, verteilte und personalisierte Informationssysteme vorgestellt. Diese wurden in Form von Tools und einem produktiv eingesetzten Konferenzplaner implementiert.

Eine Auswertung des Nutzerverhaltens des Systems soll demnächst dazu genutzt werden, dieses zu verbessern bzw. den Entwicklungsfokus auf eher bevorzugte Funktionalitäten zu richten.

Literatur

[1] P. Biswas, S. Han, and J. Wu. Location Caching in the Mobile Middleware Platform. In *Third International Conference on Mobile Data Management*. IEEE Computer Society, 2002.

[2] G. H. Forman and J. Zahorjan. The Challenges of Mobile Computing. *IEEE Computer*, 27(4), 1994.

[3] C. Lagoze and D. Fielding. Defining collections in distributed digital libraries. *D-Lib Magazine*, 1998.

[4] T. Lahiri, S. Abiteboul, and J. Widom. Ozone: Integrating structrued and semistructured data. Technical report, Stanford University, 1999. *http://www-db.stanford.edu/pub/papers/ozone.ps*.

[5] Y. L. Theng, N. Mohd-Nasir, G. Buchanan, B. Fields, and H. Thimbleby. Dynamic digital libraries for children. In *Proceedings of the First ACM/IEEE-CS Joint Conference on Digital Libraries, Roanoke, USA*, 2001.

[6] I. Witten and D. Bainbridge. *How to Build a Digital Library*. Morgan Kaufmann, 2003.

[7] A. Zeitz and I. Bruder. Object Oriented Modeling, Import, and Query Processing of Digital Documents. In C. C. Marshall, G. Henry, and L. Delcambre, editors, *ACM/IEEE 2003 Joint Conference on Digital Libraries*, 2003.