Seminar
Datenbanken und Informationssysteme-
Grundlagen webbasierter Informationssysteme

Web- und Gridservices zur Überwindung von Heterogenität bei Suche und Anfrageverarbeitung

von Lei Xia
Betreuer: Jürgen Göres
20.07.2004

Inhaltsverzeichnis:

1. Einleitung …… 3

2. Formen von Heterogenität ……... 4

3. Grundlagen …….. 6
   3.1. (Database-)Web Services …….. 6
      3.1.1. SOAP …… 7
      3.1.2. WSDL ……... 7
   3.2. (Data-) Grid ……8
      3.2.1. Virtuelle Organisationenen …..……..9
      3.2.2. Open Grid Services Infrastructure …… ……9
      3.2.3. Open Grid Services Architecture ……...……. 10
   3.3. Föderierte DBMS ……...10

4. Web Services Object Runtime Framework ... ……13

5. Grid Data Services ……..16
   5.1. Grid Data Services als Schnittstelle zu DBMS ….……...16
   5.2. Grid Data Services als Grundlagen für föderierte DBMS ….18

6. Zusammenfassung und Ausblick ……19

Literatur ……21

1. Einleitung

In den letzten Jahrzehnten haben viele Unternehmen große Investitionen zum Aufbau ihrer IT-Infrastruktur getätigt. Aufgrund der unterschiedlichen Anforderungen und der eigenen Beschäftigungssituation wurden immer die am geeignetsten erscheinenden Softwarelösung und Technologien ausgewählt, was dazu geführt hat, dass heute innerhalb vieler Unternehmen und dort sogar innerhalb einer Abteilung unterschiedliche und damit inkompatible Lösungen eingesetzt werden. Da früher jede Organisationseinheit ihre IT-Infrastruktur unabhängig von den anderen entwickelte und betrieb, stellte dies kein Problem dar. Aber mit der Entwicklung des Internet und seinem Einsatz zur Kommunikation innerhalb und außerhalb von Unternehmen bereitet diese Heterogenität zunehmend Probleme. Eine Integration der einzelnen Softwaresysteme und Datenbanken wird zunehmend erforderlich, um neue Projekte und Anwendungen zu ermöglichen oder vorhandende zu optimieren. Die innerhalb einzelner Unternehmen oder einzelner Abteilungen vorhandene Software und Hardware ist aus verschiedenen Gründen nicht kompatibel: So ist es problematisch, wenn die Systeme auf verschiedenen Plattformen realisiert wurden oder die eingesetzten Software, wie zum Beispiel Datenbanksysteme, von unterschiedlichen Anbietern stammen und inkompatible Datenmodelle haben. Die Festlegung von geeigneten Standards bietet einen möglichen Ausweg, um solche Probleme zu lösen. Alle großen Plattform- und Softwareanbieter können durch derartige Standards in ihren Produkten die Integration vereinfachen und so die Heterogenität zumindest teilweise verbergen. Webservices und Gridservices sind ein Versuch einer solchen Standardisierung und sollen im Mittelpunkt dieser Arbeit stehen. Die vorliegende Arbeit untersucht die Eignung dieser beiden eng verbundenen Technologien zur Über-windung von Heterogenität. Dazu werde ich das Produkt von IBM, das WORF genannt wird, zur Erzeugung von Web Services und die wichtigsten Technologien im (Data-)Grid vorstellen, die der Überwindung von Heterogenität insbesondere beim Zugriff auf Datenbanken und Informationssysteme dienen.

2. Formen von Heterogenität

Zunächst sollen die Formen von Heterogenität, die beim Zugriff auf Datenbanken und Informationssysteme auftreten können, vorgestellt werden [BKLW99]:

(1) Technische Heterogenität:
Hierzu gehören verschiedene technische Aspekte wie abweichende Hardware-plattformen und Betriebssysteme.

(2) Programmiersprachenheterogenität:
Verschiedene Programmiersprachen, z.B. Java, C++, etc., werden eingesetzt für die Entwicklung und Implementierung von Applikationen. Diese Programmiersprachen wurden unterschiedlich entwickelt, haben abweichende Semantik sowie Syntax und können deswegen gegenseitig inkompatibel sein.

(3) Schnittstellenheterogenität:
Der Zugriff auf Datenbanksysteme erfolgt über Programmierschnittstellen (APIs) für verschiedene Programmiersprachen. Diese Schnittstellen unterscheiden sich stark zwischen verschiedenen DBMS und erschweren somit die Anwendungsentwicklung. Für einzelne Sprachen haben sich hier bereits Standard-APIs etabliert. Zum Beispiel ist die Java Database Connectivity (JDBC) geeignet für Java.

(4) Anfragesprachenheterogenität
Die Anfragesprachen für die Suche und Manipulation von Daten in Datenbanken können zwar sehr ähnlich aber in Einzelheiten unterschiedlich und inkompatibel sein. Sie basieren zwar oft auf existierenden Standards (SQL 92/99), haben aber proprietäre Dialekte.

(5) Datenmodellheterogenität
Verschiedene Datenmodelle (hierarchisch, Entity-Relationship, relational, objekt-relational, objekt-orientiert, XML etc.) bieten verschiedene Konzepte zur Modellierung von Schemas. Viele Konzepte sind zwar in ihrer Bedeutung und Verwendungsmöglichkeit ähnlich, aber sie können nicht immer problemlos aufeinander abgebildet werden. Dadurch müssen die gleichen Sachverhalte in verschiedenen Datenmodellen unterschiedlich modelliert werden, was zur Daten-modellheterogenität führt. Außerdem bieten verschiedene Datenmodelle auch unterschiedliche Mächtigkeit zum Ausdruck einer bestimmten Eigenschaft. Beispiels-weise unterstützt das relationale Da-

tenmodell im Vergleich zum objekt-orientierten Datenmodell keine Vererbung und muss sie sehr aufwendig simulieren.

(6) Semantische Heterogenität:

Auch wenn man Schemata im gleichen Datenmodell formulieren kann, können sie wegen unterschiedlicher Auffassungen der Entwickler oder anderen Gründen zur semantischen Heterogenität führen. Im relationalen Datenmodell besteht zum Beispiel ein Schema aus Relationen mit Attributen. Deren Namen haben eine Bedeutung, die aber nicht im Schema enthalten und somit dem System nicht bekannt ist. Ein Beispiel für derartige Probleme sind Begriffe, die gleiche Namen aber je nach Domäne verschiedene Bedeutungen haben (Homonyme) oder unterschiedliche Begriffe mit identischer (Synonyme) oder ähnlicher Bedeutung (zum Beispiel Hypernyme, Hyponyme).

(7) Schemaheterogenität

Schemaheterogenität erfasst Probleme, die durch die Modellierung des gleichen Sachverhaltes mit unterschiedlichen Konzepten des verwendeten Datenmodells entstehen. Abbildung 1 zeigt ein Beispiel für einen Konflikt von „Attributname – Attributwert“. Im Beispiel enthält die linke Tabelle die Vorlesungsnamen als Attributnamen, während sie in der rechten Tabelle als Attributwerte aufgeführt sind.

Prof Name	**Logik**	**DB**	**GIM**
Maier	X		
Mueller	X	X	
Peters			X

Prof Name	**Course**
Maier	Logik
Mueller	Logik
Mueller	DB
Peters	GIM

Abbildung 1: Konflikt von „Attributname – Attributwert“

(8) Strukturelle Heterogenität

Elemente mit gleicher Bedeutung können in unterschiedlichen Datenbanken auf verschiedene Art und Weisen strukturiert werden. Zum Beispiel können gleiche Attribute in zwei Datenbanken unterschiedlicher Tabellen zugeordnet werden oder eine in Datenbank A stehende Tabelle in Datenbank B in zwei Tabellen zerlegt werden.

3. Grundlagen

Um einzelne dieser Heterogenitätsformen zu überwinden und die Integration der Daten aus verschiedenen Datenquellen zu vereinfachen, bieten Web Services eine Standardisierung des Nachrichtenaustausches (Simple Object Access Protocol, SOAP) über das Internet und eine Beschreibungssprache (Web Services Description Language, WSDL) von Schnittstellen zwischen Client und Anbieter. Im Grid werden die Datenquellen in einer virtuellen Organisation zusammengefasst, damit die Formen von Heterogenität verborgen werden können. In diesem Kapitel schildere ich kurz Web Services und Grid sowie die unterstützten Technologien. Mehr Informationen bieten die Ausarbeitungen zu den Themen „Webservices und Service Oriented Architectures“ und „Grids“ an.

3.1. Web Services

IBM definiert Web Services wie folgt:

> *„Web services is a technology that allows applications to communicate with each other in a platform- and programming language-independent manner. A web service is a software interface that describes a collection of operations that can be accessed over the network through standardized XML messaging.“*

Die Fähigkeit, wieder verwendet werden zu können, ist eine wichtige Anforderung an die Business-Dienste. Es ist ineffizient und sehr aufwendig, für verschiedene Clients immer jeweils eine Schnittstelle zu entwickeln. Web Services definieren eine Schnittstelle zwischen Client und Dienstanbieter. Sie basieren auf Standards wie HTTP, XML, SOAP, WSDL und UDDI (Universal Description, Discovery and Integration). Ein Client sucht in UDDI die erwünschten Web Services, deren Schnittstelle durch WSDL beschrieben wird, und kommuniziert mit dem Dienst über Nachrichten zum Beispiel im SOAP-Format. Web Services verbergen die konkrete Realisierung der Services gegenüber dem Client. Es wird nur der Aufbau der Nachrichten zur Kommunikation zwischen dem Client und den Anbietern festgelegt, wobei XML als textbasiertes Format plattform- und programmiersprachenunabhängig ist. Es reicht aus, eine einzige Schnittstelle für alle Clients zu entwickeln und einzusetzen, auch wenn die Clients in verschiedenen Sprachen programmiert oder auf verschiedenen Plattformen realisiert werden. Die Probleme der Formen von Heterogenität werden dadurch vermieden.

Database Web Services sind Web Services, die den Zugriff auf Datenbanken ermöglichen, wie zum Beispiel Anfrage und Manipulation von Daten. Solche Operationen können durch WSDL in standardisierter Form deklariert werden.

3.1.1. SOAP

SOAP ist ein auf XML basierendes Protokoll für den Informationsaustausch über das Internet. Es beschreibt ein standardisiertes Nachrichtenformat und ist plattform- bzw. programmiersprachenunabhängig. Jede SOAP-Nachricht hat ein *Envelope*-Element als Wurzelelement, wodurch man es als ein SOAP-Dokument erkennt. Jede Nachricht hat optional noch einen oder mehrere *Header*, die die Kontrollinformationen über die Nachricht, wie zum Beispiel eine Nachrichtennummer oder Authentisierung, enthalten. Sie muss auch einen *Body* haben, der die applikationsspezifischen Nutzinformationen, wie zum Beispiel den Aufruf einer entfernten Methode, enthält. Im Body-Element ist optional ein *Fault*-Element enthalten, das die Informationen über auftretende Fehler bei der Bearbeitung der Nachrichten bietet. SOAP kann mit mehreren unterschiedlichen Transportprotokollen HTTP, SMTP, BEEP usw.) verwendet werden. Es unterstützt viele unterschiedliche Interaktionsmuster. Neben dem klassischen Request/Response-Muster kann man auch einzelne Nachrichten verschicken oder beliebige andere Muster mit Hilfe von SOAP implementieren [KoLe04].

3.1.2. WSDL

WSDL basiert wie SOAP auch auf XML und dient dazu, die Schnittstelle eines Web Services unabhängig von der zur Realisierung verwendeten Technologien zu beschreiben. WSDL erlaubt die Deklaration von Operation und die Definition der Nachrichten, die ein Web Service empfangen kann und die er verschickt. Durch Bindings beschreibt WSDL, welche Protokolle zum Nachrichtenaustausch verwendet und wie die Nachrichten kodiert werden. Das Port-Element enthält die Adresse des Web Services. Abbildung 2 zeigt die grundlegende Struktur einer WSDL-Datei:

```
<wsdl : definitions xmlns: wsdl = “http://w3c.org/...”>
    <wsdl : documentation … />
    <wsdl : types>
        Definition der Datentypen zum Beispiel mit XML Schema
        Ein XML Schema beschreibt die Struktur eines XML-Dokuments und definiert zum
        Beispiel die Elemente im Dokument mit den zugehörigen Attributen, die Reihenfolge
        der Elemente, die Datentypen von den Elementen beziehungsweise von den Attributen usw. [MaMa99]
    </wsd : types>
    <wsdl : message>
        Definition der Nachrichten
        <wsdl : part name = … />
        Definition der in der Nachricht benutzen Typen
```

```
</wsdl : message>
<wsdl : portType>
    Eine Menge von Operationen, die der Web Service anbietet
    <wsdl : operation>
        Die Operationen mit Eingabe- und Ausgabenachrichent
        <wsdl : input message = … />
        <wsdl : output message = … />
    </wsdl : operation>
</wsdl : portType>
<wsdl : binding>
    Die Kodierung der Nachrichten und die Protokolle zum Nachrichtenaustausch
</wsdl : binding>
<wsdl : service>
    Kollektion mehrerer WS
    <wsdl : port name = …  binding = …>
        Ein Endpoint, das einer Kombination von einem Binding mit einer Netzwerkadresse dient
        <soap : address location = …/>
    </wsdl : port>
</wsdl : service>
</wsdl : definitions>
```

Abbildung 2: Die Grundstruktur einer WSDL-Datei

Durch die WSDL-Beschreibung erfährt der Client direkt, wo er den Dienst findet und wie er mit ihm kommunizieren kann. Wie erwähnt ist diese Beschreibung unabhängig von der Implementierung der Dienste, auch wenn der Client und der Anbieter unterschiedliche Programmiersprachen und Technologien benutzen.

3.2. Grids

Ein Grid ist eine Infrastruktur von Hardware und Software, die aus dem Wunsch heraus entstand, verteilt vorliegende Hochleistungsrechner, Software, Daten und andere Ressourcen, die teilweise ungenutzt bleiben, über schnelle Datenleitungen konsistent und möglichst kostengünstig zu integrieren. Auf diese Weise sollen extrem speicherplatz- und rechenzeit-intensive Probleme effizient mit vorhandener Hardware gelöst werden. Im Idealfall erscheint ein Grid aus Sicht des Anwenders oder Entwicklers wie ein einziger Rechner. Die wichtigsten Merkmale eines Grid-Systems sind lokale Autonomie, Heterogenität der Ressourcen und fehlende zentrale Kontrolle.

Auch verteilt vorliegende und in heterogenen Systemen gespeicherte Daten können als Ressourcen aufgefasst werden. Ein Grid, das den Zugriff auf solche Daten ermöglicht, wird als Data Grid benannt. Die Institutionen, die auf eine große Menge von geographisch verteilten Daten zugreifen, sie analysieren und verarbeiten wollen, benötigen Unterstützung durch eine Infrastruktur, die Standard-APIs für den Zugriff anbieten und die Zusammensetzung der Daten aus unterschiedlichen Datenbanken ermöglicht. Das Data Grid ist ein spezialisiertes, erweitertes Grid und bietet eine solche Möglichkeit. Außer den generellen Grid Services bietet es noch spezialisierte Services bezüglich der Daten wie zum Beispiel den Zugriff auf Metadaten-Repositories und auf die Speichersysteme.

3.2.1. Virtuelle Organisationen

Ein im Bereich Grid häufig anzutreffender Begriff ist die virtuelle Organisation (VO). Eine virtuelle Organisation setzt sich aus einer Menge von Individuen aus möglicherweise verschiedenen Institutionen zusammen, die ein gemeinsames Ziel verfolgen und zu diesem Zweck für die Dauer der Kooperation ihre Ressourcen bündeln [FOKT01]. Dazu gehören nicht nur Ressourcen wie Rechenleistung und Speicherplatz, sondern insbesondere auch die verteilt vorliegende heterogene Daten, um damit die großen Herausforderungen in Industrie und Wissenschaft anzugehen. Grid-Technologie ermöglicht den Aufbau und die Verwaltung einer solchen VO. Es gibt eine Middleware, die mehrere Datenquellen zu einer einzelnen virtuellen Datenquelle zusammenfasst, so dass es für den Client im Idealfall erscheint, als ob es nur eine Datenbank mit einem Datenmodell, klarer Semantik auf einer Plattform und einem Betriebssystem gäbe. Wenn der Client eine Anfrage stellt, sucht die VO die zur Beantwortung nötigen Datenquellen, überträgt an diese die jeweiligen Teile der Anfrage und gibt anschließend die integrierten Ergebnisse zurück. Der Client muss nicht wissen, wie die erwünschten Dienste kombiniert werden und die Anfrage bei Datenanbietern lokal ausgeführt wird.

3.2.2. Open Grid Services Infrastructure

Alle Grid Services sind Web Services, die sich zusätzlich an eine Menge von Konventionen und Standards halten und mit einer erweiterten Form von WSDL, der Grid-WSDL (GWSDL), beschrieben werden. Open Grid Services Infrasturcture (OGSI) definiert ein Mechanismus zur Erzeugung und Verwaltung der Grid Services als zustandbehaftete Web Services. Die OGSI-Standards legen einige Routineoperationen fest, die jeder Grid Service unterstützt. Dadurch wird die Instanzierung (*Factory*-PortType), Auffindung beziehungsweise Referenzierung (*FindServiceData, GSH, GSR*) von Services realisiert, der Verfügbarkeitszeitraum (*lifetime*) von Serviceinstanzen verwaltet und die Zustandsveränderung asynchron benachrichtigt (*Notification*). Nach den OGSI-Standards werden die auftretenden Fehler auch einheitlich behandelt. Die Grid Services können einen Zustand

besitzen, der durch *Service Data Elements* beschrieben wird. Sie können, ähnlich einer Vererbungshierarchie in der objekt-orientierten Programmierung, als Erweiterung von anderen Services beschrieben werden.

3.2.3. Open Grid Services Architecture

Ein wichtiger Punkt zur Realisierung eines Grids ist die Standardisierung der Schnittstellen, so dass alle Ressourcen in einer standardisierten Form entdeckt, integriert und überwacht werden. OGSA nimmt genau diese Rolle ein.

OGSI spezifiziert Grid Services. Die von OGSI definierten Standardschnittstellen dienen lediglich zur Vereinheitlichung gewisser Grundfunktionalitäten. Basierend auf den OGSI-Grundkonzepten definiert Open Grid Services Architecture (OGSA) tatsächlich nutzbare konkrete Schnittstellen für anwendungsorientierte Dienste. Dazu gehören Ressourcenverwaltung, Ausführungsverwaltung, Datendienste (Data Services), Sicherheits- und Informationsdienste. Im Grid werden alle diese Schnittstellen standardisiert durch GWSDL beschrieben. Im Rahmen dieser Ausarbeitung betrachten wir später den Grid Data Services (siehe. Kapitel 5), eine von OGSA definierte Schnittstelle und in virtueller Organisation eingesetzt wird, um ein Standard-API anzubieten und damit die API-Heterogenität zu lösen.

3.3. Föderierte DBMS

Föderiertes DBMS [RNC+03] sind eine etablierte Technologie, um Informationen aus mehreren eventuell heterogenen Datenquellen zu kombinieren. Als eine Middleware bietet ein föderiertes DBMS einen standardisierten transparenten Zugriff auf verteilte, heterogene Datenquellen durch das Verbergen von unterschiedlichen Plattformen, Datenmodellen, Schemata, Strukturen, Anfragesprachen und Schnittstellen. Für die Clients eines solchen Systems erscheint es im Idealfall als ein einziges herkömmlichen Datensystem.

Abbildung 3 zeigt die typische Architektur eines föderierten DBMS. Die Clients können Standardschnittstellen, wie zum Beispiel JDBC oder ODBC, für den Zugriff auf das föderierte DBMS genauso nutzen, wie diese auch für einfache relationale DBMS genutzt werden. Das föderierte DBMS ist unabhängig von der Implementierung der Datenquellen und fungiert als Middleware, die mit den heterogenen Datenquellen über sogenannte Wrapper kommuniziert. Ein Wrapper ist ein Menge von Codes und dient als Schnittstelle zur Kompatibilität. In föderierten DBMS kann ein Wrapper bei Bedarf dynamisch geladen werden. Föderierte DBMS verbergen die Formen von Heterogenität zwischen verschiedenen Datenquellen und bietet dem Client eine integrierte virtuelle Datenbank und

damit ein Datenmodell, ein Schema sowie eine Struktur an. Zur Bearbeitung einer von Client gestellten Anfrage identifiziert das föderierte DBMS die jeweils relevanten Datenquellen und entwickelt einen Anfrageverarbeitungsplan zur Abbildung der Anforderung des Client auf die physischen Datenquellen, welche die Anfrage verarbeiten sollen. Der Plan zerlegt normalerweise die Anfrage in mehrere Fragmente zur Ausführung in den jeweiligen Datenquellen sowie zur Ausführung durch das föderierte DBMS für das Zusammenführen der Ergebnisse von den Datenquellen. Da die Wrapper die Anpassung

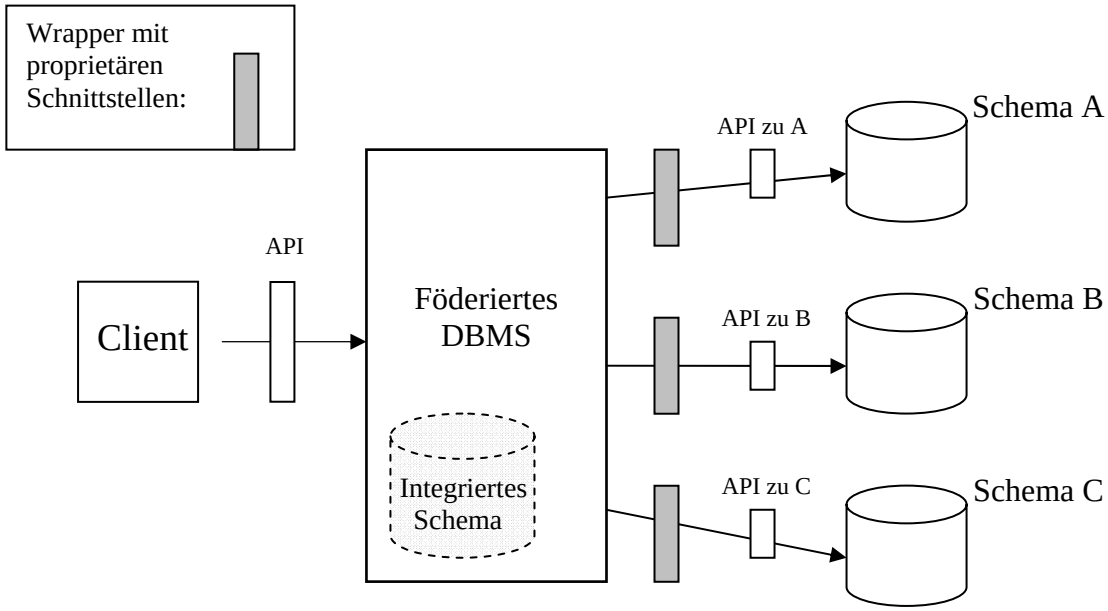

Abbildung 3: Architektur eines föderierten DBMS

an die Schnittstelle des föderierten DBMS vornehmen, kann ein föderiertes DBMS auch auf ursprünglich nicht vorgesehene Quellen zugreifen, da für diese lediglich ein zusätzlicher Wrapper zur Verfügung gestellt werden muss. Normalerweise kann ein Wrapper für den Zugriff auf mehrere Datenquellen genutzt werden, solange diese die gleiche Schnittstelle (API) benutzen. In der Realität setzen Datenquellen von verschiedenen Herstellern jedoch meist unterschiedliche APIs ein. Daher sind für die unterschiedlichen Quell-APIs spezialisierte Wrapper erforderlich. Außerdem sind für unterschiedliche föderierte DBMSs auch unterschiedliche APIs zwischen Wrapper und föderierten DBMS notwendig. Das Konzept des föderierten DBMS kann auf Data Grids übertragen werden. Ein Föderationsdienst, dessen interne Realisierung möglicherweise auf einem bestehenden föderierten DBMS basiert, nutzt statt eines speziellen APIs zu den Wrappern ein generisches API, das in einer sprachenunabhängigen Weise beschrieben wird (zum Beispiel als Web Services mit WSDL oder als Grid Services mit GWSDL). Somit kann dieses föderierte DBMS alle Datenquellen, die dieses generische API unterstützen und dabei in einer beliebigen Sprache auf unterschiedlichen Plattformen implementiert sein können, zu ei-

ner virtuellen Datenbank integrieren. (siehe Kapitel 5.2. Grid Data Services als Grundlagen für föderiertes DBMS)

4. Web Services Object Runtime Framework

Web Services können als plattform- und programmiersprachenunabhängige Zugriffsmöglichkeit auf Datenquellen genutzt werden. Verschiedene Hersteller von Datenbanksystemen haben Produkte entwickelte, um ihre Systeme als Web Services zugänglich zu machen. Im folgenden Kapitel soll das Produkt von IBM, das Web Services Object Runtime Framework (WORF), vorgestellt werden.

WORF bietet eine Umgebung zur einfachen Erzeugung von Web Services für den Zugriff auf IBM DB2-Datenbanksystem. WORF benutzt DADX-Dokumente (Document Access Definition Extension) zum Spezifizieren der Web Services.

DADX ermöglicht die kombinierte Beschreibung von Schnittstelle und Implementierung von Web Services für DB2. Aus einer DADX-Datei kann WORF die WSDL-Schnittstellenbeschreibung automatisch generieren. Eine DADX-Datei beschreibt eine Menge von Operationen, die durch SQL-Statements (inklusive Stored Procedures mit Input- und Outputparametern) und DAD-Dateien definiert werden, wobei Stored Procedures mehrere SQL-Statements zusammenfassen und als eine einzige Web-Service-Operation benutzt werden können. Für Web Services, welche die Generierung von XML-Daten aus oder die Speicherung von XML-Daten in DB2 bieten, wird der DB2 XML Extender benutzt. Es gibt zwei Möglicheiten zum Speichern von XML-Daten: entweder speichert DB2 XML Extender das ganze XML-Dokument in DB2 bzw. in einem File-System oder er bildet durch XML-Kollektionsmethode die XML-Daten in relationalen Daten ab und speichert sie in DB2. DAD-Daten (Document Access Definition) beschreiben diese Abbildung zwischen XML-Daten und relationalen Daten.

Zwei Arten von Operationen werden von DADX-Dateien unterstützt:

1. SQL-Operationen:
   Drei Operationstypen werden hier definiert:
   <query> : Anfrage auf den Daten in einer DB2-Datenbank
   <update> : Veränderung von Daten in der Datenbank wie zum Beispiel insert, delete und update
   <call>: Aufruf von Stored Procedures

2. XML-Kollektionsoperationen:
   Es gibt zwei Operationstypen:
   <retrieveXML>: Generierung der XML-Dokumente durch Komposition der existierenden relationalen Daten

<storeXML> : Speichern der XML-Dokumente als relationale Daten in DB2

Ein Beispiel einer DADX-Datei sieht wie folgt aus (Abbildung 4):

```
<?xml version="1.0" encoding="UTF-8"?>
  <DADX
    xmlns=”http://schemas.ibm.com/db2/dxx/dadx”
    xmlns:wsdl=”http://schemas.xmlsoap.org/wsdl/ ”>
    <operation name="listDepartments">
      <query>
        <SQL_query> select* from order_tab
                    where customer_name = :customer_name
        </SQL_query>
        <parameter name="customer_name" type="xsd:string"/>
      </query>
    </operation>
  </DADX>
```

Abbildung 4: Beispiel einer DADX-Datei

Die Attribute xmlns und xmlns:wsdl im Wurzelelement DADX spezifizieren die Namensräume von DADX und WSDL. Das Beispiel enthält eine Web-Service-Operation mit dem Namen “listDepartments”. Diese Operation führt eine SQL-Anfrage mit einem Parameter im SELECT-Statement aus.

Abbildung 5 zeigt, wie WORF einen “Web Service Request” bearbeitet.

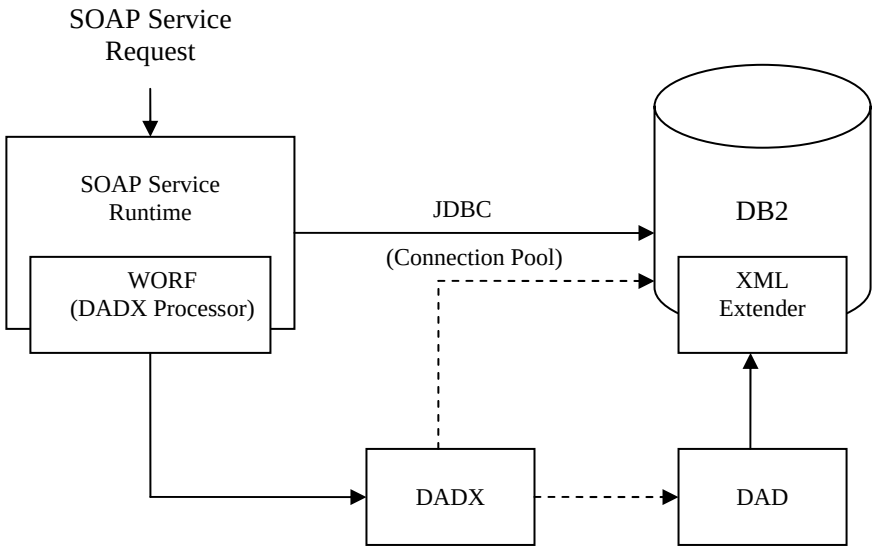

Abbildung 5: WORF und XML Extender

WORF empfängt die Web-Service-Anfrage, die den Namen der Datenressourcen und die aufgerufenen Operationen des Web Services enthält. Darauf lädt WORF die entsprechende DADX-Datei des Dienstes und bei XML-Operationen auch die zugehörige DAD-Datei. Dann verbindet sich WORF über JDBC mit der DB2 und führt die Operationen durch. Die Ergebnisse werden in XML-Daten transformiert und als SOAP-Nachricht zurück zum Client geschickt.

WORF ist eine J2EE-Applikation und auf mehreren Betriebssystemen installierbar: Windows NT, Windows 2000, Linux, AIX und Solaris. Als Applikationsserver, auf dem WORF läuft, wird Websphere Application Server oder Apache Web Server benutzt.

WORF schafft eine Brücke zwischen Clients und Dienstanbietern. Es ermöglicht die Beschreibung des Zugriffs auf Daten in DB2 durch XML-Dokumente und kann WSDL-Beschreibungen automatisch generieren. Dadurch wird die Kommunikation zwischen den Clients und den Dienstanbietern in einer standardisierten Form ausgeführt. Der Einsatz von DB2 XML Extender und DAD bietet eine Abbildung zwischen XML-Daten und relationalen Daten und ermöglicht dadurch auch die Interoperationen zwischen den beiden Datenbanken mit unterschiedlichen Datenmodellen. Der Nachteil von WORF ist die starke Abhängigkeit von SQL und JDBC. Außerdem arbeit WORF nur mit IBM DB2.

5. Grid Data Services

OGSA definiert Schnittstellen für zahlreiche Konzepte in einem dienstbasierten Grid. Im Rahmen dieser Ausarbeitung konzentrieren wir uns auf die Data Services. Die Data Services sind Web Services und ermöglichen eine Verbindung zwischen Client und Datenanbietern ohne dem Client die physische Lokalisierung der Daten sichtbar zu machen. Sie bieten als standardisierte Schnittstelle eine Zugriffsmöglichkeit auf strukturierte Datenquellen an. Ähnlich wie funktional vergleichbare WS-Schnittstellen definieren Grid Data Services eine Abstrahierung von den technischen Aspekten der Datenquellen und verbergen damit die Plattform- und Programmiersprachenheterogenität, d.h. der Zugriff auf die Datenbanken erfolgt auf standardisierte Weise und ist unabhängig von ihrer tatsächlichen Implementierung. WSDL wird hier auch benutzt, damit alle Dienstschnittstellen in einer standardisierten Form beschrieben werden können.

5.1. Grid Data Services als Schnittstelle zu DBMS

Ein Data Service ist ein Web Service, der mindestens eine der in Abbildung 6 beschriebenen Basisschnittstellen implementiert und damit die Verwaltung und den Zugriff auf Datenressourcen ermöglicht [FTU+04]. Alle Basisschnittstellen sind standardisiert durch WSDL beschrieben. Wir können Grid Data Services wie Web-Service-Schnittstelle auch als eine Standardschnittstelle zu DBMS einsetzen.

Datenvirtualisierung ist ein wichtiger Begriff in den Grid Data Services. Eine von einem Data Service implementierte Datenvirtualisierung ist eine abstrahierte Sicht von Daten aus einer oder mehreren Datenquellen. Zum Beispiel können wir einen Data Service definieren, der ein JPEG-Image in einer relationalen Datenbank als ein File oder als eine Menge von relationalen Tabellen zur Verfügung stellt, um verschiedene Zwecke zu erreichen. Eine Datenvirtualisierung kann direkt aus einer entsprechenden Datenressource stammen oder durch Transformation des Datenformats realisiert werden. Sie kann auch aus einer Teilmenge von Daten aus einer Datenressource abstrahiert werden. Mehrere Datenressourcen können auf eine Datenvirtualisierung abstrahiert werden. Eine Datenressource kann auch auf mehreren Datenvirtualisierung abgebildet werden.

PortType	Beschreibung
DataDescription	Beschreibt den Inhalt und die Struktur von Data Service.
DataAccess	Bietet Operationen für die Anfrage und Manipulation der Daten in einer Datenvirtualisierung.
DataManagement	Bietet Operationen für die Überwachung und Verwaltung der Datenvirtualisierung von einem Data Service.
DataFactory	Bietet Operationen für die Erzeugung eines neuen Data Services.

Abbildung 6: Vier Basisdatenschnittstellen (WSDL PortTypes)

Abbildung 7 zeigt die Architektur von Data Service [FTU+04]:

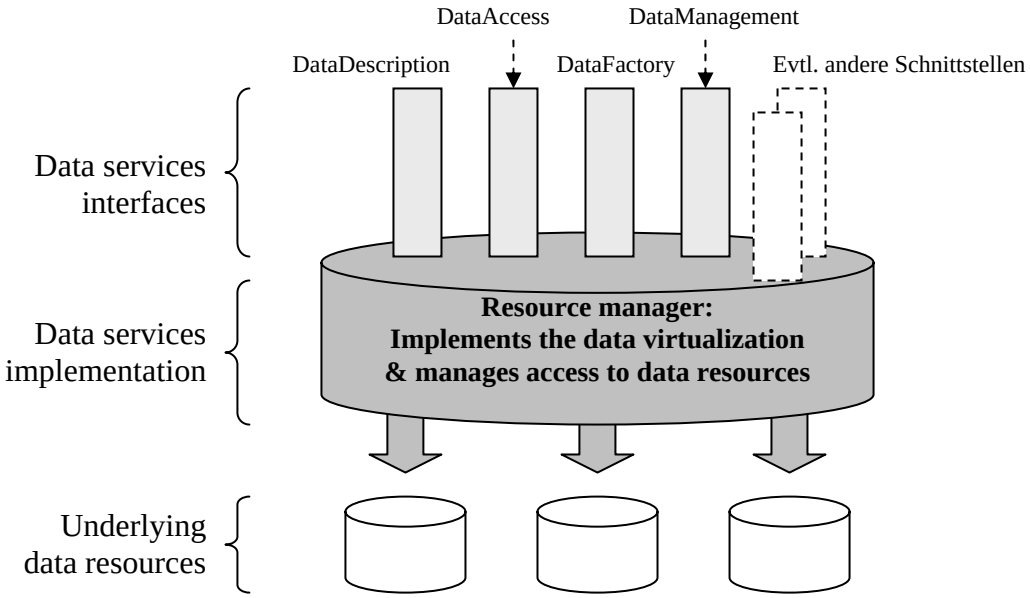

Abbildung 7: Architektur von Data Service

Ein Beispiel bietet Abbildung 8 [AAK+04]. Der Database Data Service hat eine RelationalDescription-Schnittstelle, die auf einer Basisschnittstelle zur Beschreibung des Inhalts von Database Data Service, der DataDescription, basiert. Der SQLAccess-PortType ermöglicht die Interaktionen mit der relationalen Datenbank,damit die SQL-Operationen vom Database Data Service implementiert und die Ergebnisse direkt zurück zum Client geschickt werden können. Der Client kann auch die createService-Operation aufruft, um einen vom Database Data Service abstammenden Rowset Data Service zu erzeugen. Die RowsetAccess-Schnittstelle bietet den Zugriff auf Row set an, das aus der Durchführung eines SQL-Statements resultiert und eine Teilmenge von Daten aus relationalen Datenbank enthält. Es wird representiert in einer Form von Tabellen.

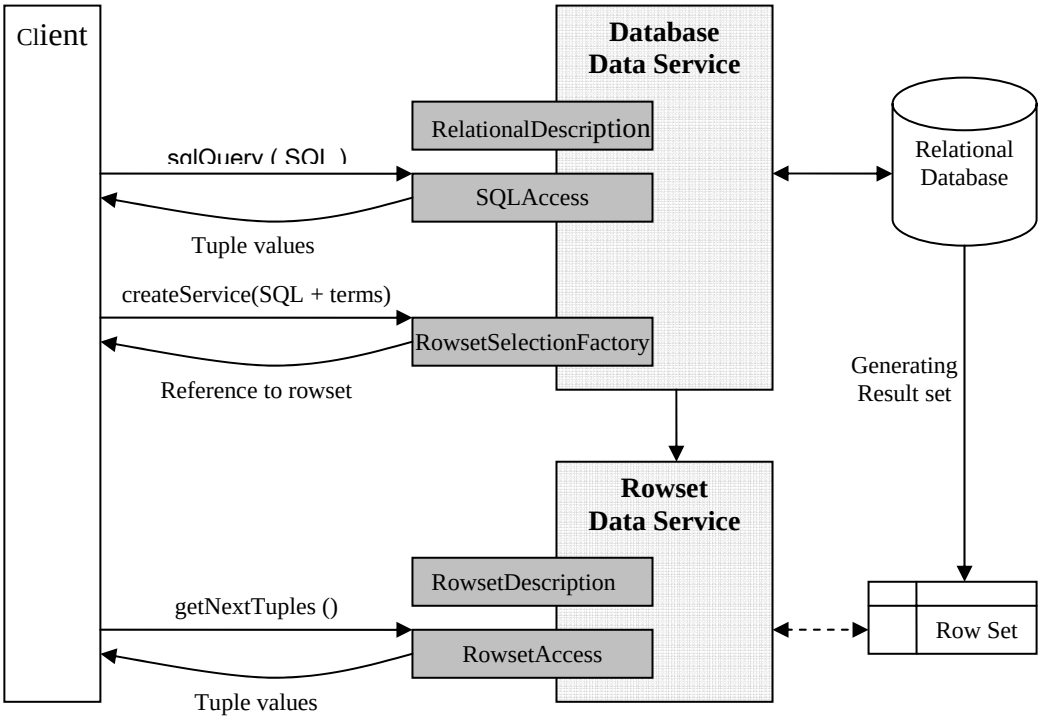

Abbildung 8: Beispiel für einen Grid Data Service

Die DataAccess-Schnittstelle bietet Operationen für den Zugriff auf Datenvirtualisation in einem Data Service. Diese Operationen werden durch WSDL beschrieben, damit man die Data Services als Standardschnittstellen zu DBMS benutzen kann. Es gibt mehrere Erweiterungen von DataAccess-Schnittstelle, die die *GGF DAIS working group* und andere Gruppen definieren kann, zum Beispiel ist SQLAccess eine solcher Erweiterung und bietet SQL-basierte Anfragen und Manipulationen auf relationalen Daten. Diese Schnittstelle kann noch für unterschiedliche SQL-Dialekte erweitert werden, die von verschiedenen Datenbankprodukten unterstützt werden. XMLCollectionAccess ist eine andere Verfeinerung von dieser Basisschnittstelle. Sie ermöglicht den XPath-, XQuery und XUpdate-basierenden Zugriff auf XML-Daten und funktioniert analog zu SQLAccess.

5.2. Grid Data Services als Grundlagen für föderierte DBMS

Im Grid können Grid Data Services als Standardschnittstelle für föderierte Services und Datenquellen benutzt werden. Wie im Kapitel 3.3 erwähnt, verbirgt ein föderiertes DBMS mehrere Formen von Heterogenität zwischen verschiedenen Datenquellen, aber es erfordert auch hohen Aufwand. Für unterschiedliche Quell-APIs müssen jeweils spezialisierte Wrapper genutzt werden. Außerdem nutzen gegenwärtige föderierte DBMS proprietäre Wrapper-Schnittstellen und somit sind die Wrapper im Allgemeinen nur für ein einziges föderiertes DBMS-Produkt geeignet. Wenn ein neues föderiertes DBMS erfoder-

lich ist, müssen auch neue Wrapper entwickelt werden, was sehr aufwendig und ineffizient ist. Deswegen könnte alternativ ein sprach- und systemunabhängiges Standard-Interface (zum Beispiel in Form eines Grid Data Services) genutzt werden. Dadurch kann ein auf einem föderierten DBMS basierender föderierter Service im Grid über eine standardisierte Schnittstelle auf verschiedene Datenquellen zugreifen und den Clients die integrierte virtuelle Datenbank durch die gleiche Schnittstelle anbieten. Wenn ein neuer föderierter Service eingesetzt wird, der mit Grid Dat Services umgehen kann, muss kein neuer Wrapper entwickelt werden (Abbildung 9).

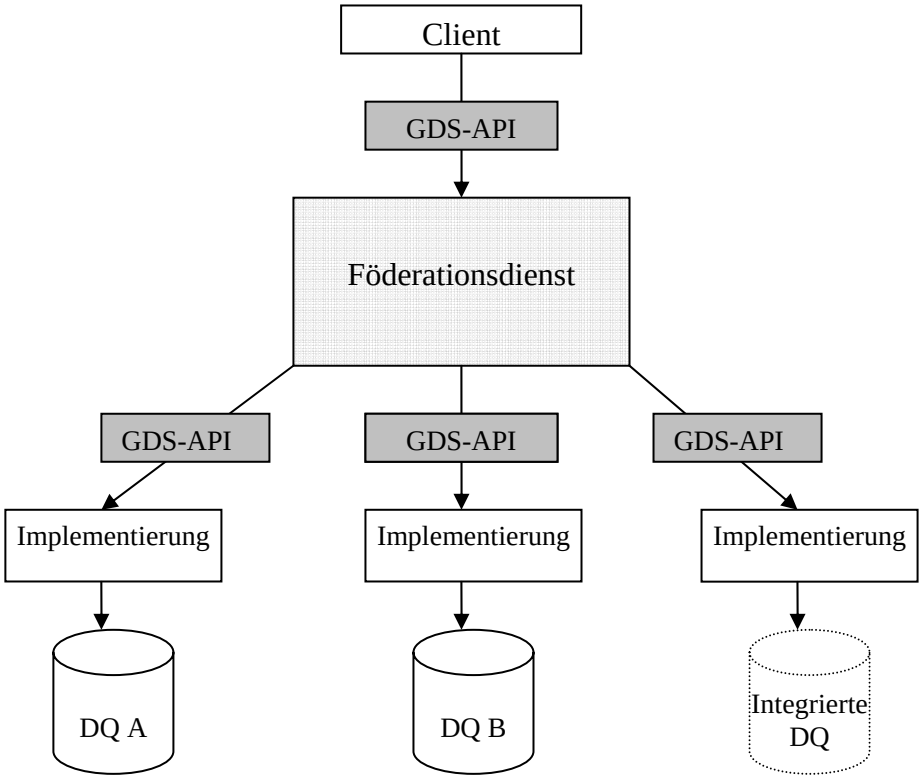

Abbildung 9: Föderationsdienst mit Grid Data Service-API

Im Abbildung 9 sehen wir zudem, dass auch eine Hierarchie von Föderationsdiensten unterstützt wird. Auf der unteren Ebene kann man durch zusätzliche Föderationsdienste und standardisierte APIs (zum Beispiel Grid Data Services) mehrere Datenquellen auf eine virtuelle Datenquelle integrieren und die als normale Datenquelle auf der höheren Ebene wieder mit anderen Datenquellen integrieren.

6. Zusammenfassung und Ausblick

Web Services benutzen SOAP zum Informationsaustausch über das Internet und WSDL zur Beschreibung der Schnittstellen, wobei beide Standards auf XML basieren. XML ist plattform- und programmiersprachenunabhängig. Solche Standards ermöglichen den Web Services, die konkrete Implementierung der Dienste gegenüber dem Client zu verbergen. Grid Data Services bauen auf Web Services auf und bieten ein Standard-API für den Zugriff auf verteilt vorliegende, autonome und heterogene Datenquellen. Wie normale Web Services benutzen sie die auf WSDL basierende Standard-Beschreibungssprache GWSDL für die Beschreibung der Operationen von Grid Services. Diese Standardisierungen von Web- und Grid Services bieten eine Lösung für die Form von technischer Heterogenität, Programmiersprachenheterogenität und API-Heterogenität zwischen Clients und Dienstanbietern an. Das Konzept von föderierten DBMS ermöglicht, eine integrierte virtuelle Datenbank aufzubauen und die Heterogenitäten zwischen verschiedenen Datenquellen teilweise zu verbergen.

Wir sehen aber auch, dass diese Technologien keine vollständige Lösung für alle auftretenden Formen von Heterogenität darstellen. Zum Beispiel kennen Web Services die Bedeutung der Elemente in Datenquellen nicht und können deswegen die semantische Heterogenität nicht überwinden. Dies muss manuell erfolgen. Sie bieten auch keine standardisierte Anfragesprache für den Zugriff auf verschiedene Datenquellen. SQL ist eine Basissprache für die Anfrage auf relationale Daten, kann aber für verschiedene DBMS in Einzelheiten unterschiedlich sein. Außerdem bleibt die Datenmodellheterogenität erhalten. Diese Probleme sollen in der Zukunft gelöst werden, zum Beispiel soll es eine allgemein uniforme Programmierschnittstelle entwickelt oder eine Abbildung zwischen verschiedenen Datenmodellen ermöglicht werden.

Alle Grid Dienste können Zustandsinformationen besitzen. Die Zustandinformationen werden durch *Service Data* mit GWSDL in PortType beschrieben. Außerdem ermöglicht GWSDL auch die Beschreibung von Vererbung der PortType. Auch für allgemeine Web Services werden Zustandsinformationen benötigt, was durch WS-Resource Framework (WSRF) realisiert werden kann. WSRF ist ähnlich zur OGSI-Spezifikation, benutzt aber unterschiedliche Syntax und Terminologie. Mit Hilfe von *Resource property elements* beschreibt WSRF die Zustandsinformationen von Web Services. Die W3C Web Services Description Working Group beschäftigt sich zur Zeit mit WSDL 2.0, die viele Funktionalitäten von GWSDL übernimmt.

Literatur

HaBr03 H. Haas, A. Brown,
Web services glossary
W3C Working Draft, August 2003,
http://www.w3c.org/TR/2003/WD-ws-gloss-20030808

Mitr03 N. Mitra,
SOAP version 1.2 part 0: primer
2003,
http://www.w3c.org/TR/2003/REC-soap12-part0-20030624

CCM+01 E.Christensen, F. Curbera, G. Meredith, S. Weerawarana,
Web services description language (WSDL) 1.1
2001,
http://www.w3c.org/TR/wsdl

KoLe04 D. Kossmann, F. Leymann,
Web Services
Informatik Spektrum, 26 April 2004

FoKi04 I. Foster, H. Kishimoto,
The Open Grid Services Architecture, Version 1.0
2004,
http://www.ggf.org/ogsa-wg/

BKL+99 S. Busse, R. Kutsche, U. Leser, H. Weber,
Federated Information Systems: Concepte, Terminology and Architectures
April 1999

FKN+02 I. Foster, C. Kesselman, J. Nick, S. Tuecke,
The Physiology of the Grid-- An Open Grid Services Architecture for Distributed Systems Integration
2002

AAK+04 M. Antonioletti, M. Atkinson, A. Krause, S. Laws,
Grid Data Service Specification
15 March 2004

RNC+03 V. Rahmen, I. Narang, C. Crone, L. Haas, S. Malaika, T. Mukai, D. Wolfson, C. Baru,
Services for Data Access and Data Processing on Grids
9 February 2003

FoKT01 I. Foster, C. Kesselman, S. Tuecke,
The Anotomy of the Grid
2001

TCF+03 S. Tuecke, F. Czajkowski, I. Foster, J. Frey, S. Graham, C. Kesselman, T. Maquire, T. Sandholm, D. Snelling, P. Vanderbilt,
Open Grid Services Infrastructure (OGSI), Version 1.0
27 June 2003

Fost02 I. Foster,
What ist he Grid? A Three Point Checklist
20 July 2002

ReSc04 A. Reinefeld, F. Schintke,
Grid Services-Web Services zur Nutzung verteilter Ressourcen
Informatik Spektrum, 26 April 2004

FTU+04 I. Foster, S. Tuecke, J. Unger, A. Luniewski,
OGSA Data Services
24 February 2004

MaMa9 A. Malhotra, M. Maloney,
XML Schema Requirements
15 February 1999,
http://www.w3.org/TR/NOTE-xml-schema-req.html#Principles