A Generic Database Web Service for the Venice Lightweight Service Grid

Michael Koch

Bachelorarbeit

Michael Koch
University of Kaiserslautern, Germany
Integrated Communication Systems Lab

Email: m_koch2@cs.uni-kl.de

Agenda

- Einführung und Motivation
  - Grids und Datenbanksysteme
  - Besondere Merkmale des Dienstes
- Basistechnologien
- Architektur des Datenbankdienstes
- Evaluation – ein Testszenario
- Zusammenfassung

Motivation – Grids und Datenbanksysteme

- Gridsysteme benötigen zuverlässige und performante Datenbanksysteme
  - Viele verschiedene lose gekoppelte Applikationen arbeiten zusammen
  - Techniken für uniformen Zugriff existieren (JDBC, ODBC, etc.)
  - Nachteile: hohe Komplexität, proprietäres Treibermanagement, Firewalls
- Neue Anforderungen und Konzepte notwendig
  - Zugriff auf heterogene Datenbanksysteme mit Hilfe eines gemeinsamen Standards
  - Nutzung von beliebigen Clientsystemen, plattform- und programmiersprachenunabhängig
  - Beachtung von domänenübergreifenden Sicherheitaspekten

Motivation – Grids und Datenbanksysteme

- Venice, ein leichtgewichtiges Servicegrid
  - Framework für die vereinfachte Entwicklung, Integration und Installation verschiedenster verteilter Dienste
  - Single Sign-On, Informationbroker, SOA-Konzepte
  - Viele Anwendungen benötigen Zugang zu relationalen DBMS
- Vorhandene Technologien
  - Beispielsweise OGSA-DAI für das Globus Toolkit
  - Zugriff auf verschiedene Datenquellen unter Beachtung von Sicherheitsaspekten
  - Steuerung mit speziellen XML-Dokumenten
  - Rückgabe in kanonischem XML-Format

Motivation – Besondere Merkmale

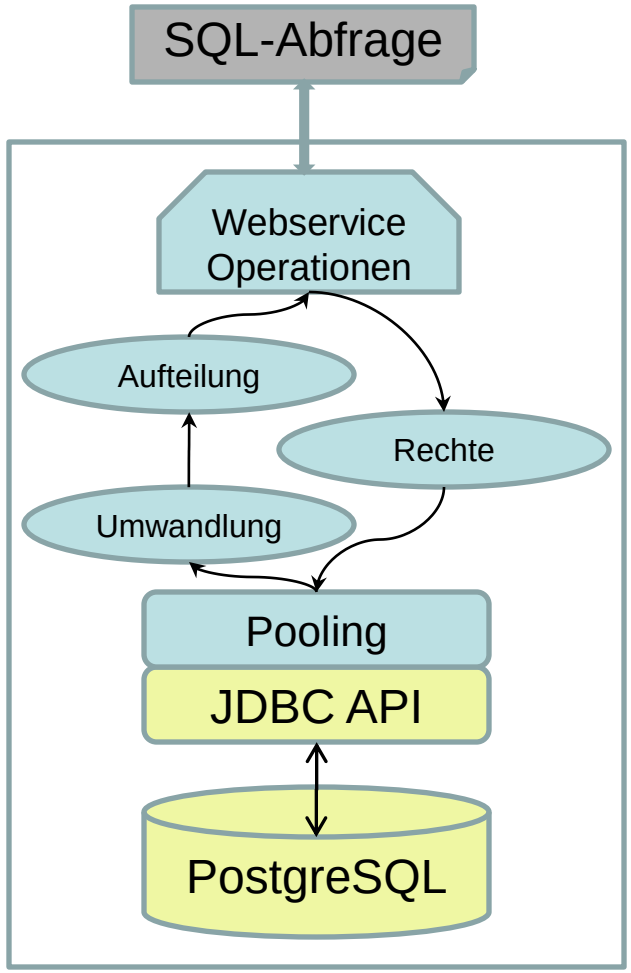
- Generischer Datenbankdienst
  - Neuer Dienst innerhalb einer Veniceföderation
  - Abstrahiert vom zugrundeliegenden DBMS und bietet einen standardisierten Zugriff an
- Besondere Merkmale
  - Fest definierte Schnittstelle mit stark typisierten Datentypen
  - SQL-Abfragen werden direkt in SOAP-Nachrichten eingebettet
  - Aufteilung von SOAP-Nachrichten und parallele Übertragung
  - Intuitive clientseitige Nutzung durch Nachbildung der JDBC API
  - Zugriffskontrolle mit Hilfe des Venice-SSO-Mechanismus

Basistechnologien

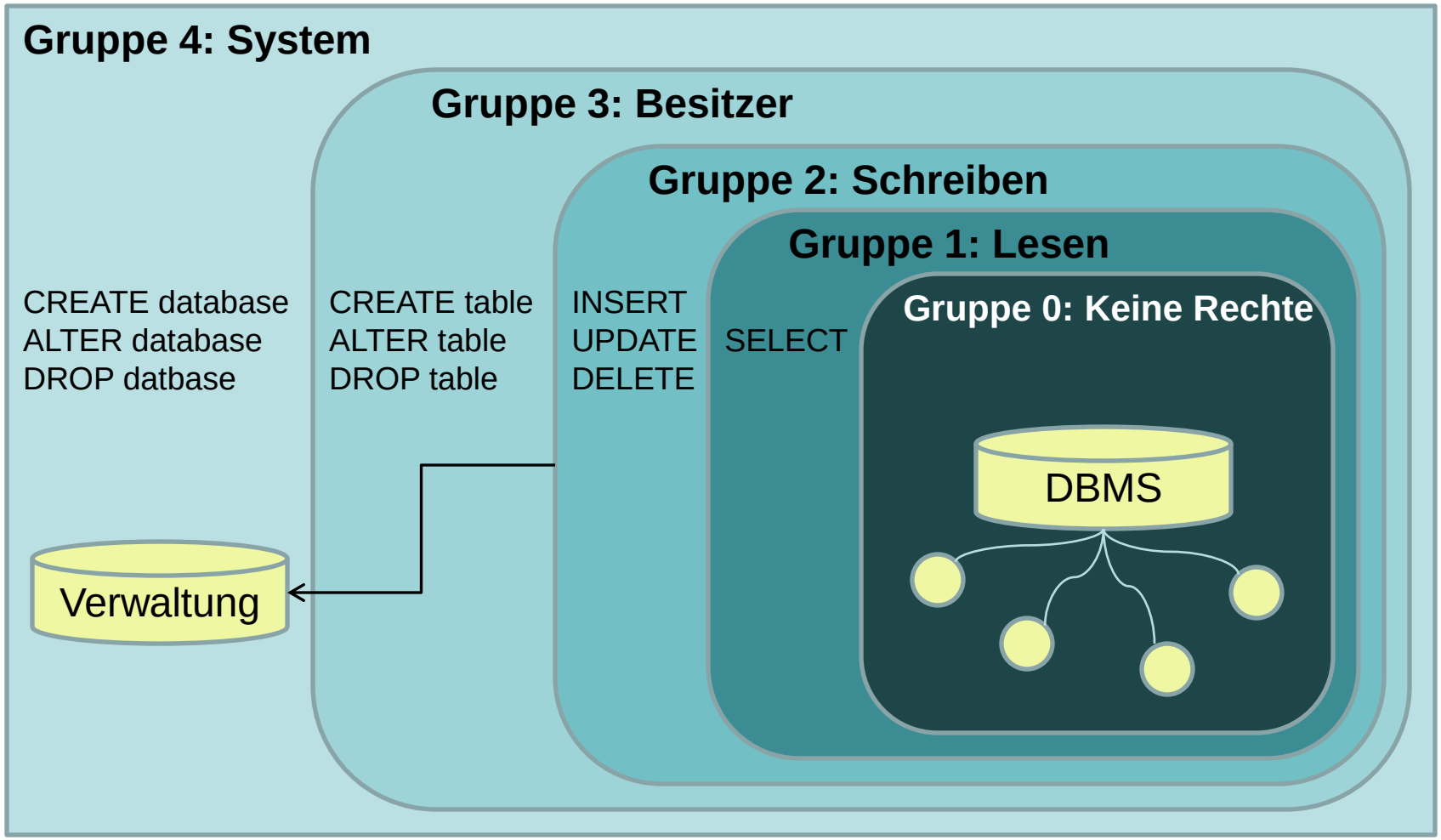
- Serviceorientierte Architekturen
  - Wiederverwendbarkeit, Dienst-Kontrakt, Lose Kopplung, Abstraktion, Zusammensetzbarkeit, Autonomie, Zustandslosigkeit, Auffindbarkeit
- Webservices
  - Kombination mehrere XML-basierter Technologien (Datentypen über XML-Schemata)
  - Schnittstelle: WSDL – Web Service Description Language
  - Transfer: SOAP-Nachrichten
- JAX-RPC (de-)serialisiert für Java automatisch; komplexe Datentypen als Javabeans
- Zugriff auf DBMS mit JDBC API

Architektur – eine Abfrage

1. SQLQuery-Objekt wird an den Datenbankdienst gesendet
2. Rechtesystem überprüft Abfrage mit regulären Ausdrücken
3. JDBC-Verbindung
4. Typumwandlung
5. Aufteilung der Nachricht
6. Übertragung



Architektur - Rechtesystem



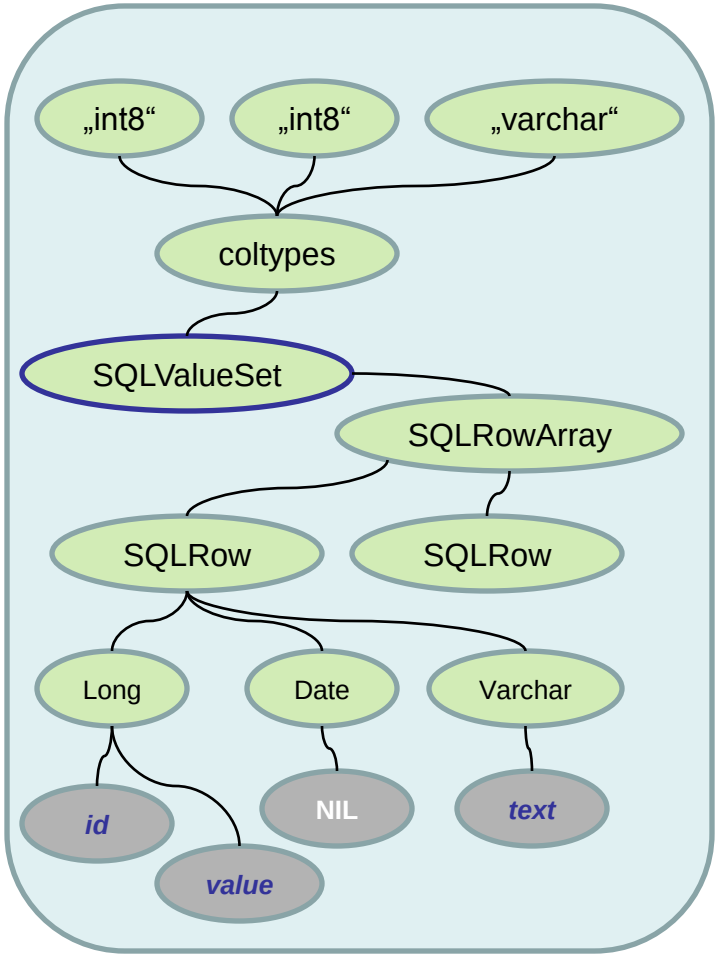
Architektur - Datentransfer

XML-Schema

```
<complexType name="SQLRow"><sequence>
<element name="int8" type="basic:LongArray" nillable="true"/>
<element name="date" type="basic:DateArray" nillable="true"/>
<element name="varchar" type="basic:StringArray" nillable="true"/>
…
</sequence></complexType>
…
<complexType name="SQLValueSet„><sequence>
<element name="colTypes" type="tns:colType“/>
<element name="SQLRows" type="tns:SQLRowArray”/>
…</sequence></complexType> ...
```

JAX-RPC

SOAP-Nachricht



SQL-Abfrage

```
SELECT * FROM example;
```

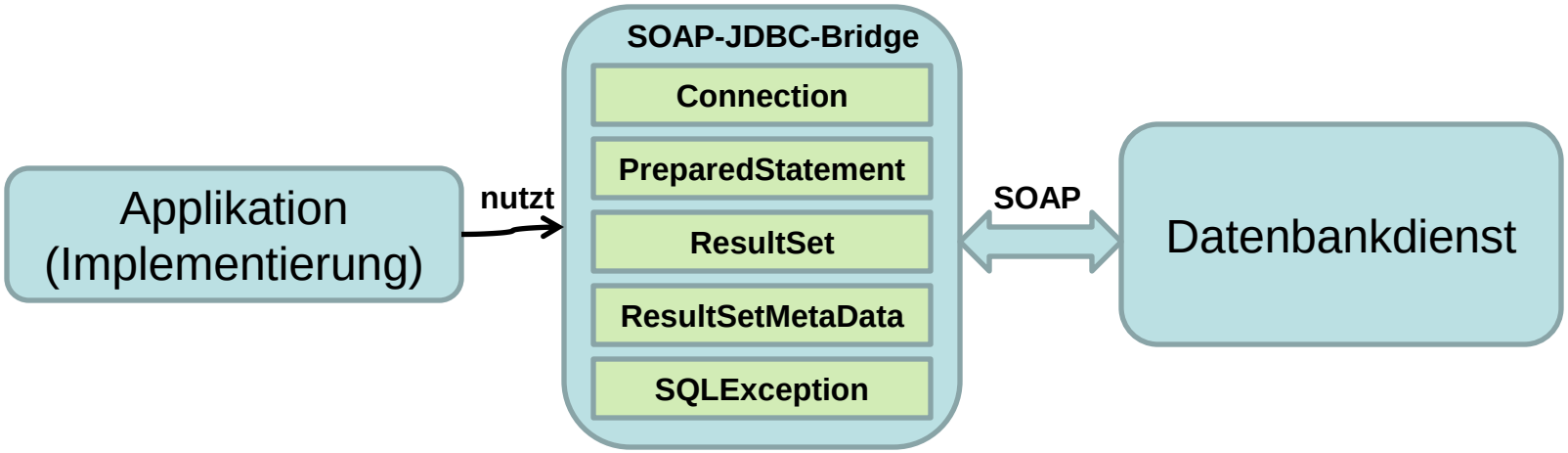
```
TABLE example (
        Id serial,
        value bigint,
        text varchar(150),
        …
);
```

JDBC API

Metadaten

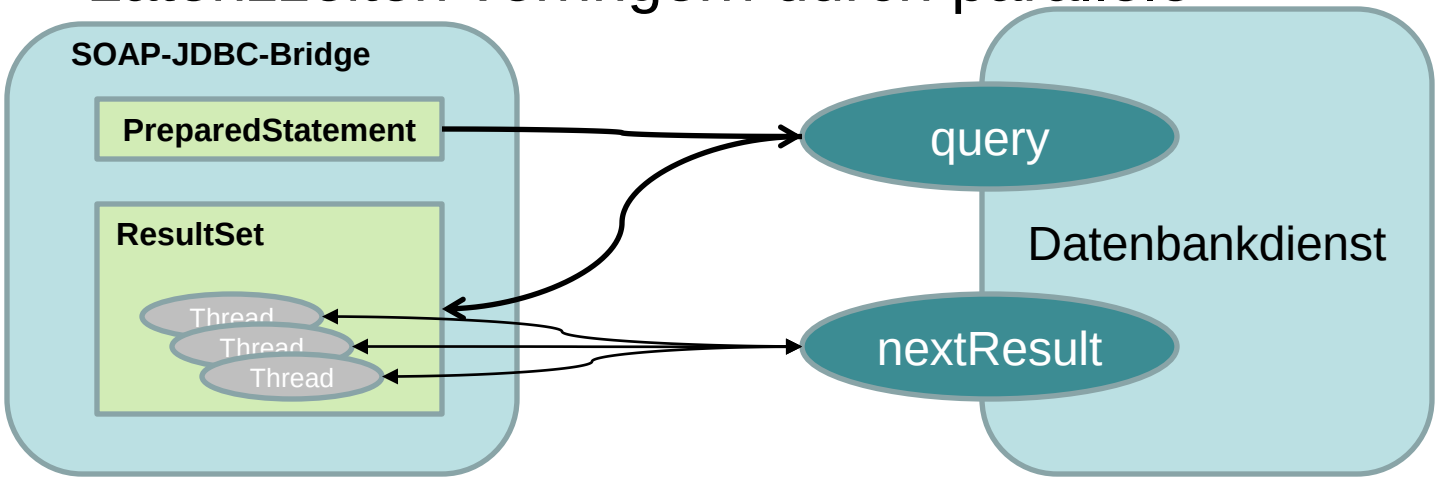
Architektur – Wrapperklassen

- Komplizierte, verschachtelte Strukturen
- ➢ Keine intuitive Nutzung und schwierige Adaption von vorhandenen Komponenten
- ➢ Einführung SOAP-JDBC-Bridge als Wrappersystem



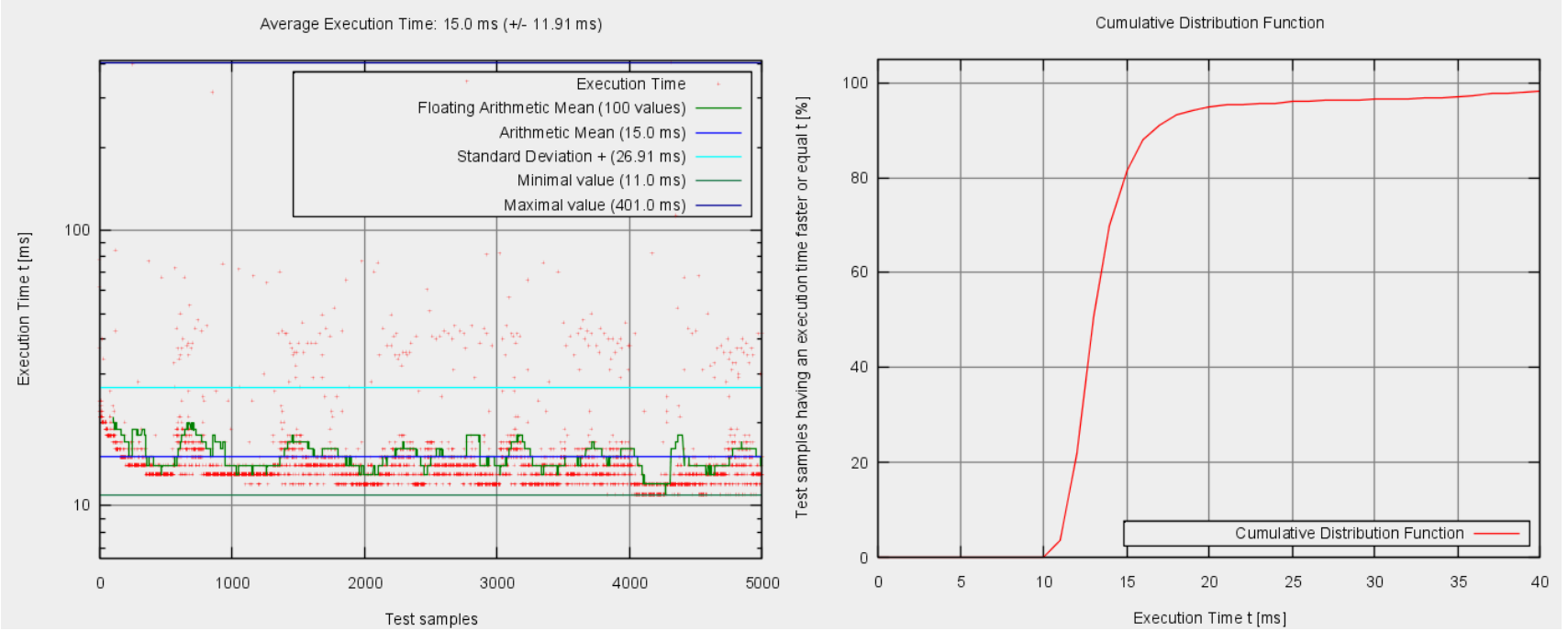
Architektur - Nachrichtenaufteilung

- SOAP-Nachrichten werden schnell sehr groß
- Aufteilung von SOAP-Nachrichten in mehrere Teile
  - Dateigrößenbeschränkungen: Vorausberechnung der Nachrichtengröße
  - Zwischenspeicherung der Ergebnisse
  - Latenzzeiten verringern durch parallele



Evaluation – ein Testszenario

Ausführungszeit eines Insert-Statements



- 14 Spalten mit unterschiedlichen Datentypen
- Durchschnittliche Verarbeitungszeit von 15 ms, davon 11 ms Übertragungszeit
- Über 95% der Anfragen werden unter 20 ms ausgeführt
- JDBC benötigt etwa 3 ms für die Bearbeitung

Zusammenfassung

- Implementierung einer generischen Datenbankschnittstelle, die von allen Anwendungen einer Veniceföderation genutzt werden kann
- Verknüpfung von Venice-SSO und SQL-Abfragen
- starke Typisierung beim Datentransfer
- Einfache Nutzung und Adaption aufgrund JDBC-ähnlicher Wrapperklassen
- Aufteilung und parallele Übertragung von großen Datenmengen