Global Info – Technisches Arbeitsreffen / Hamburg

Technische Grundlagen einer prototypischen Realisierung

Kai-Uwe Sattler   Stefan Conrad   Martin Endig   Eike Schallehn

Fakultät für Informatik

Otto-von-Guericke-Universität Magdeburg

{kus|conrad|endig|eike}@iti.cs.uni-magdeburg.de

2. Dezember 1999

FRAQL

- FRAQL– Federation Query Language
- Anfragesprache für (objektrelationale) Datenbankföderationen
- Erweiterung von SQL um
  - Datenbankföderationen
  - benutzerdefinierte Datentypen
  - Zugriff auf Metadaten
  - Restrukturierungsmechanismen
  - Konfliktbehandlung
- Ziele:
  - Erweiterbarkeit von Anfragesprachen
  - Optimierungstechniken
- Verwandte Arbeiten: SchemaSQL, DataJoiner, Mediatoren ...

FRAQL: Konzepte

- **Objekttypen**
  - Strukturbeschreibung von Tupeln
  - Spezialisierungshierarchie

```
create type type_name [ under type_names ] (
  attrib_definition
);
```

- **Datenquellen**
  - lokale Datenbank mit Relationen
  - Datenbankadapter, Verbindungsinformationen

```
register source source_name at 'DSN=db;UID=user;PWD=password'
  using 'adaptor_name';
```

FRAQL: Konzepte

- **Objektrelationen**
  - *Importrelation:* Projektion über lokale Relation

```
create table global_name of type_name
  as import from source.local_name
  [ mapping_definitions ];
```

  - *Integrationsrelation:* Sicht über globale Relationen

```
create table global_name of type_name
  as table_expression;
```

FRAQL: Konzepte

- **Benutzerdefinierte Funktionen** (*stored functions*)
  - Implementierung als statische Methoden in Java
  - dynamische Registrierung
  - Definition inverser Funktionen

```
create function func_name (fraql_type)
  returns fraql_type
  external name 'class_name.method_name'
  [ as inverse to func_name ];
```

FRAQL: Konzepte

- **Verfeinerung der Verbund- und Vereinigungsoperationen**
  - **`union`**: Vergleichsattribute

```
table1 union table2 on attr1, attr2
```

  - **`join`**: Vergleichsausdruck und Auflösungsfunktion

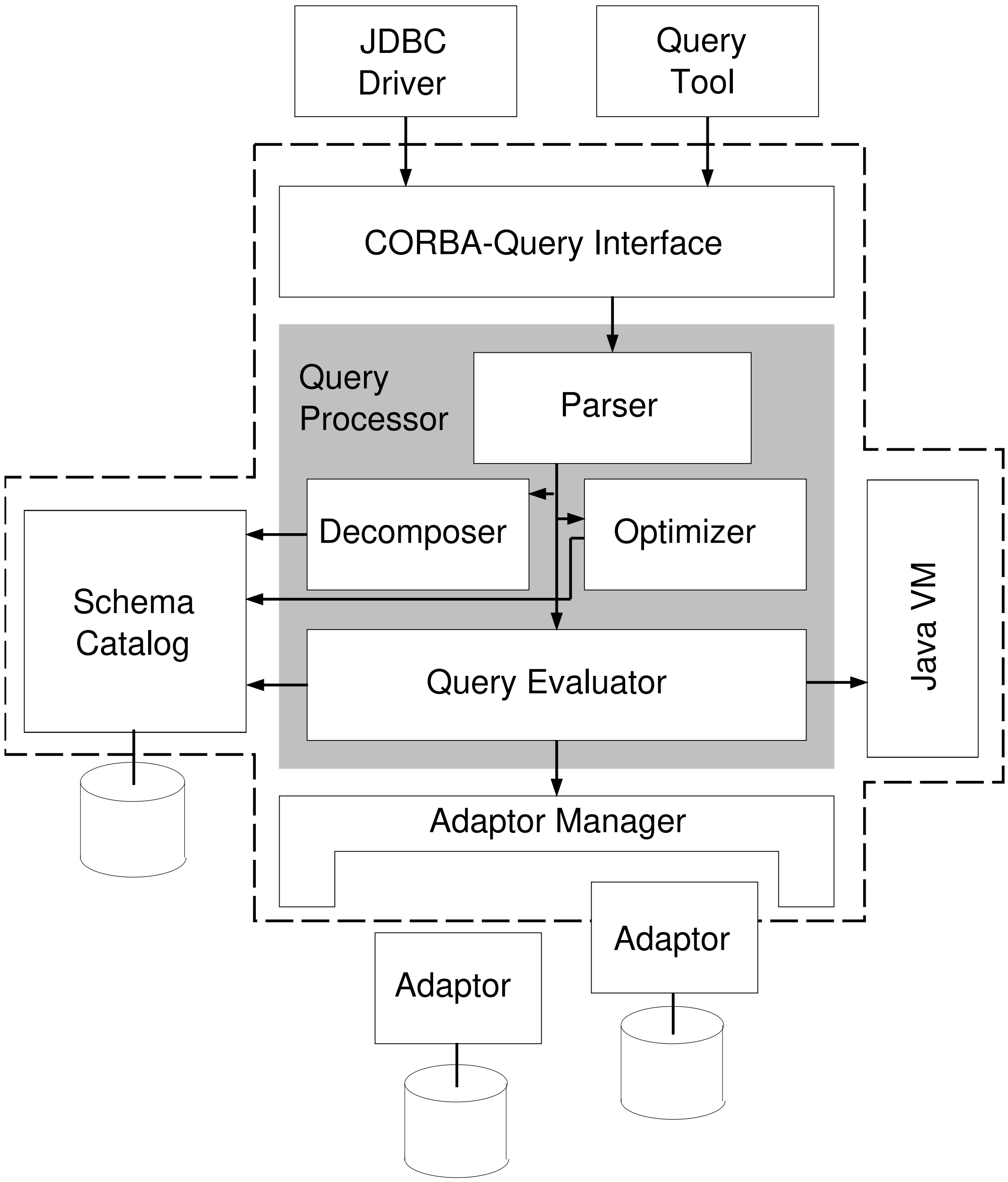
```
table1 join table2 on table1.attr1 = table2.attr1
table1 join table2 on attr1 = attr2 reconciled by func
```

FraQL: Konzepte

- **Restrukturierung von Relationen**
  - „Umordnung“ von Relationen ähnlich SchemaSQL
  - Zugriff auf Metadaten (Schemakatalog):
    `catalog.tables`, `catalog.columns`, Benutzerrelationen
  - Variablensubstitution in Anfragen:
    Berechnung der Namen von Relationen oder Attributen
    Bsp.: `tbl1.$(tbl2.col)`

```
select t2.isbn, t2.title
from catalog.tables t1, $(t1.name) t2
where t1.type_name = 'book';
```

FRAQL: Architektur



Beschreibungskonflikte

Auflösung durch Definition von Import-Relationen

1. Definition eines globalen Typs
2. Abbildung zwischen lokaler Relation und Typ
   - ➠ Umbenennung von Attributen
   - ➠ Konvertierung von Attributwerten durch benutzerdefinierte Funktionen

```
create type book (
  isbn varchar(20),
  title varchar(100),
  price float
);
```

```
create table german_books of book
  as import from src.buch (
  title is titel,
  price is dm2dollar (preis)
);
```

Semantische Konflikte

Auflösung durch Integrationsoperationen in globalen Sichten

➠ **Tupelidentität:** Definition der Vergleichsattribute

books1	*isbn*	*author*	*title*
	382578	Williams, T.	Otherland
	326523	Gibson, W.	Idoru

book2	*isbn*	*author*	*title*
	382578	Tad Williams	Otherland
	276830	St. Lem	Solaris

```
create table books of book as
  books1 union books2 on isbn;
```

Semantische Konflikte

- **Datenkonflikte:** Auflösungsfunktion zur Behandlung von Konflikten in „gleichen“ Tupeln
  - Festlegung einer Vorzugsrelation
  - Mittelwert- oder Summenbildung

bstore1	*isbn*	*number*
	3324524	2
	6710767	2

bstore2	*isbn*	*number*
	3324524	1
	1267894	2

```
create table stores of book as
  bstore1 union bstore2
  on isbn reconciled by book_resolve;
```

stores	*isbn*	*number*
	3324524	3
	1267894	2
	6710767	2

Strukturelle Konflikte

- **Partitionierung:**
  - lokale Relationen repräsentieren verschiedene Aspekte der globalen Relation bei gleichzeitiger Existenz semantisch äquivalenter Attribute
  - Auflösung durch Verbundoperationen (θ-Join, Outer-Join) und benutzerdefinierte Auflösungsfunktionen (**`reconciled by`**)

- **Meta-Konflikte:**
  - unterschiedliche Repräsentation von Konzepten (Schemaobjekt $\Leftrightarrow$ Datenobjekt)
  - Auflösung durch Restrukturierung und Nutzung von Metadaten

Meta-Konflikte: Relation vs. Attribut

db1.publ	*id*	*title*	*type*
	382660	Databases	book
	017062	Computer	journal

db2.book	*id*	*title*
	382660	Databases
	361556	Java

db2.journal	*id*	*title*
	017062	Computer
	000102	CACM

```
create table publication of publ_type as
  select t2.id, t2.title, t1.name
  from catalog.tables t1, $(t1.name) t2
  where t1.schema = 'db2';
```

Meta-Konflikte: Attribut vs. Datenwert

db1.book	*isbn*	*price*	*supplier*
	38934237	59.00	meyers
	38934237	48.00	jones

db2.book	*isbn*	*meyers*	*jones*
	38934237	59.00	48.00
	22660513	27.00	29.00

```
select b.isbn, b.$(a.name), a.name
from db2.book b, catalog.attributes a
where a.table = 'book' and a.name <> 'isbn';
```

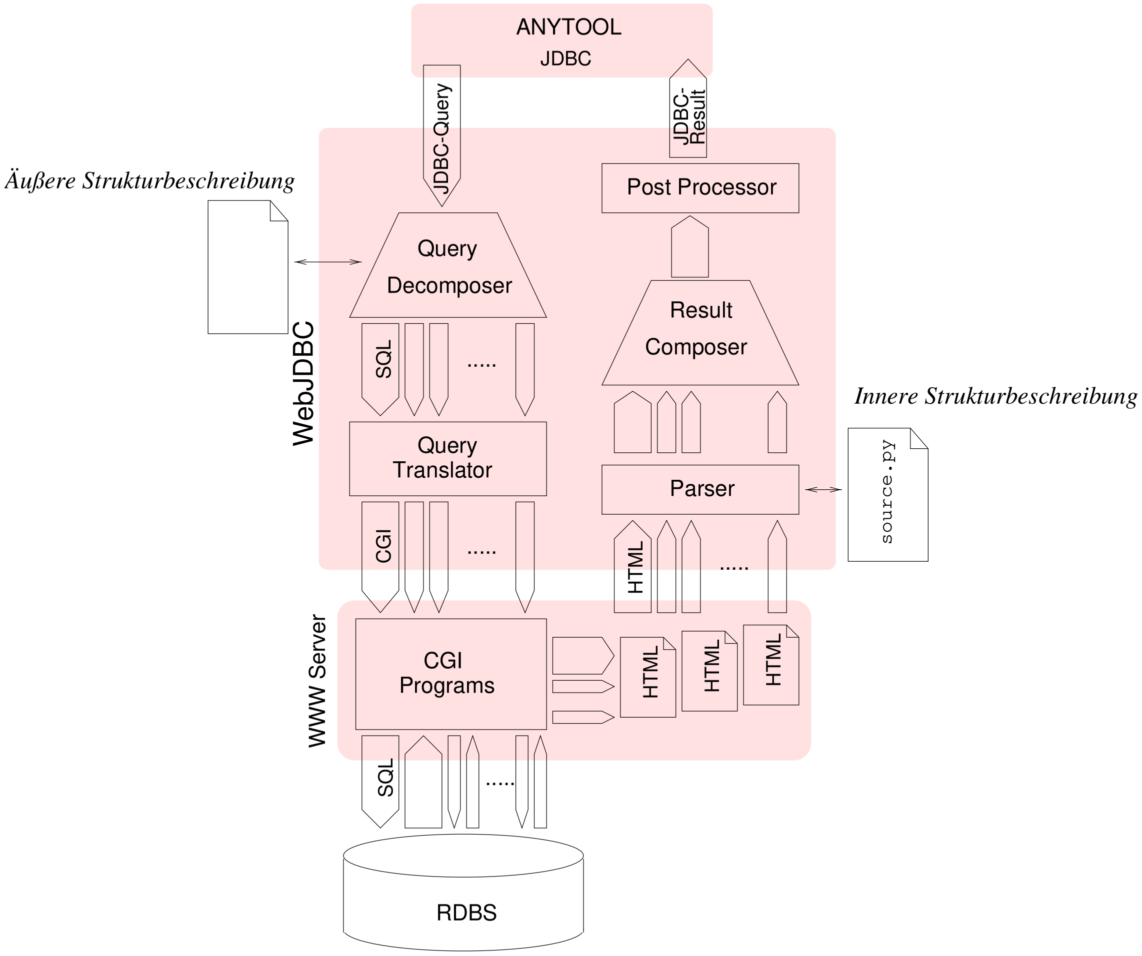
FraQL Implementierung

- Implementierung als Anfrageprozessor für lose gekoppelte Datenbankföderationen
  - Basis: CORBA, C++
  - ca. 12.000 LOC
  - einfache heuristikbasierte Optimierung
  - integrierte Java VM für benutzerdefinierte Funktionen
- Adapter
  - dynamisch nachladbar
  - für Oracle und MySQL
- Schnittstellen zum Anfrageprozessor
  - CORBA-Query Service
  - JDBC
- Zugriff auf Web-Quellen über WebJDBC

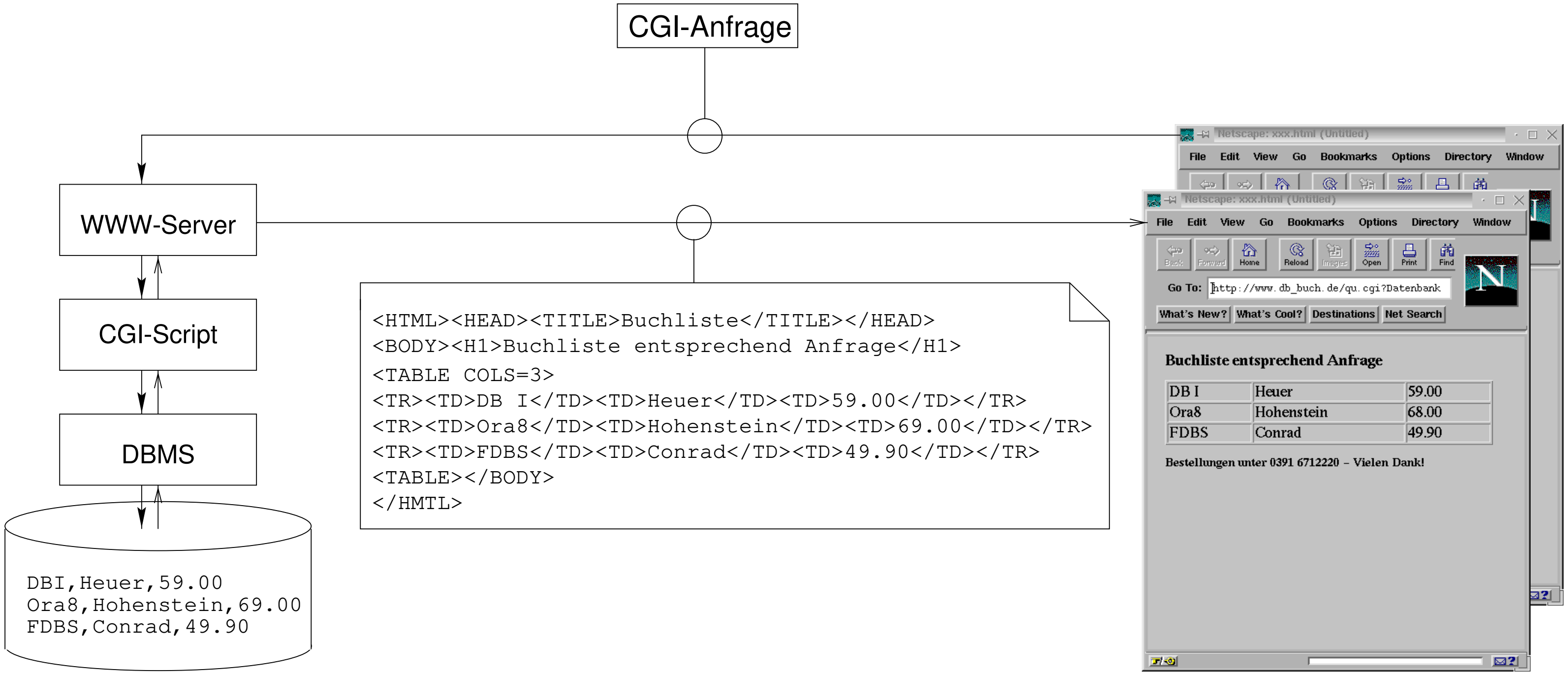
WebJDBC - Extraktionsvorschrift

```
def s(authors)
  elements=split(authors,'; ')
  return elements

def t(extension)
  titel,authors,tmp3,isbn,price=extract(tupleString,
   'SIZE=1> <BR><B>$1:</B><I>$2;
   </I><BR>$_tmp3;<BR>ISBN$3,<BR>DM$4')
  return title,map(s,authors),isbn,price

def root(htmlText)
  extension=split(htmlText,', OES')
  return map(t,extension)
```

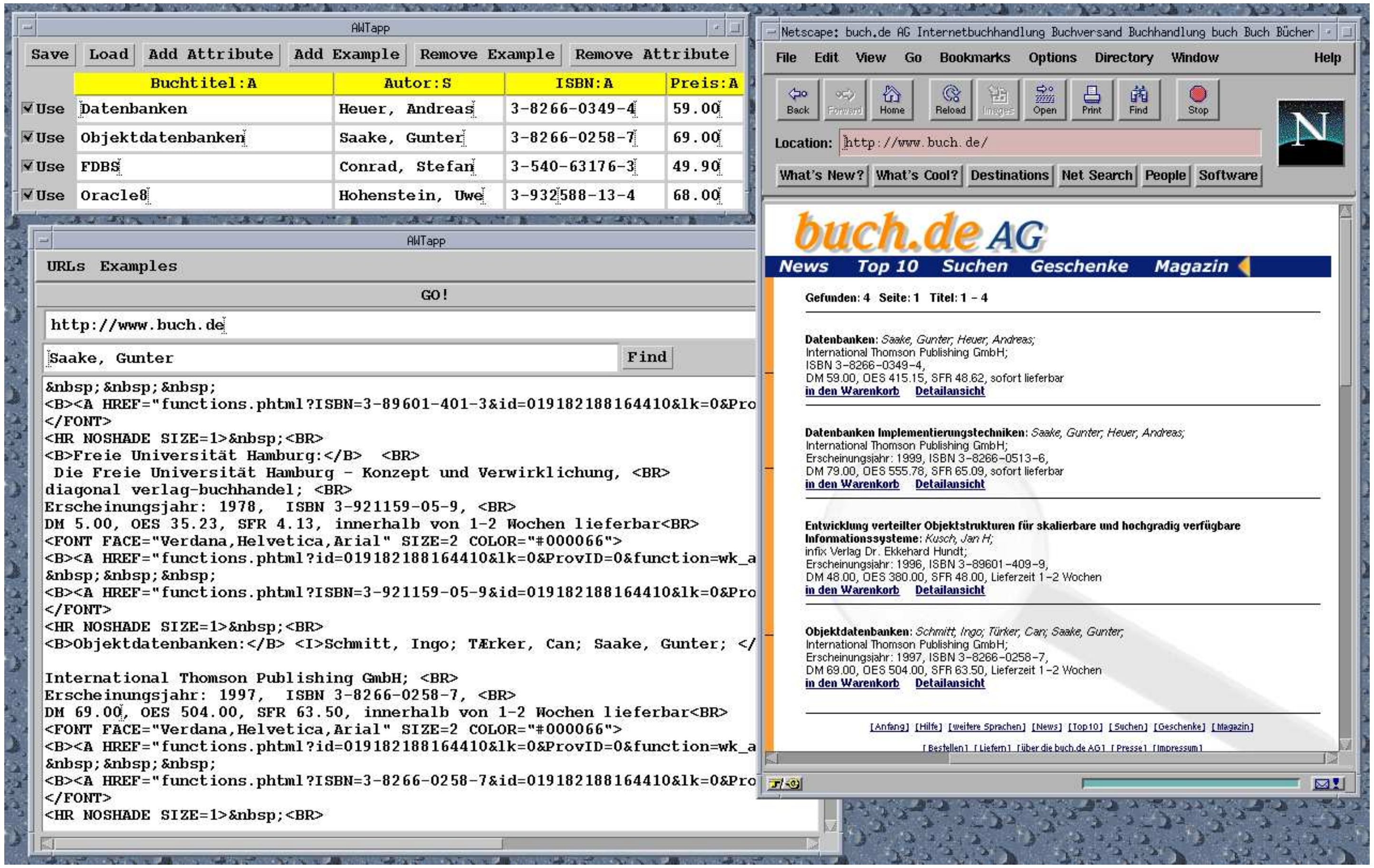
WebJDBC – Architektur



WebJDBC – Grundprinzip



Toolsupport für Adaptergenerierung



Geplante Prototyparchitektur