Kapitel 6

Wrapper-basierte Integration

Inhalt

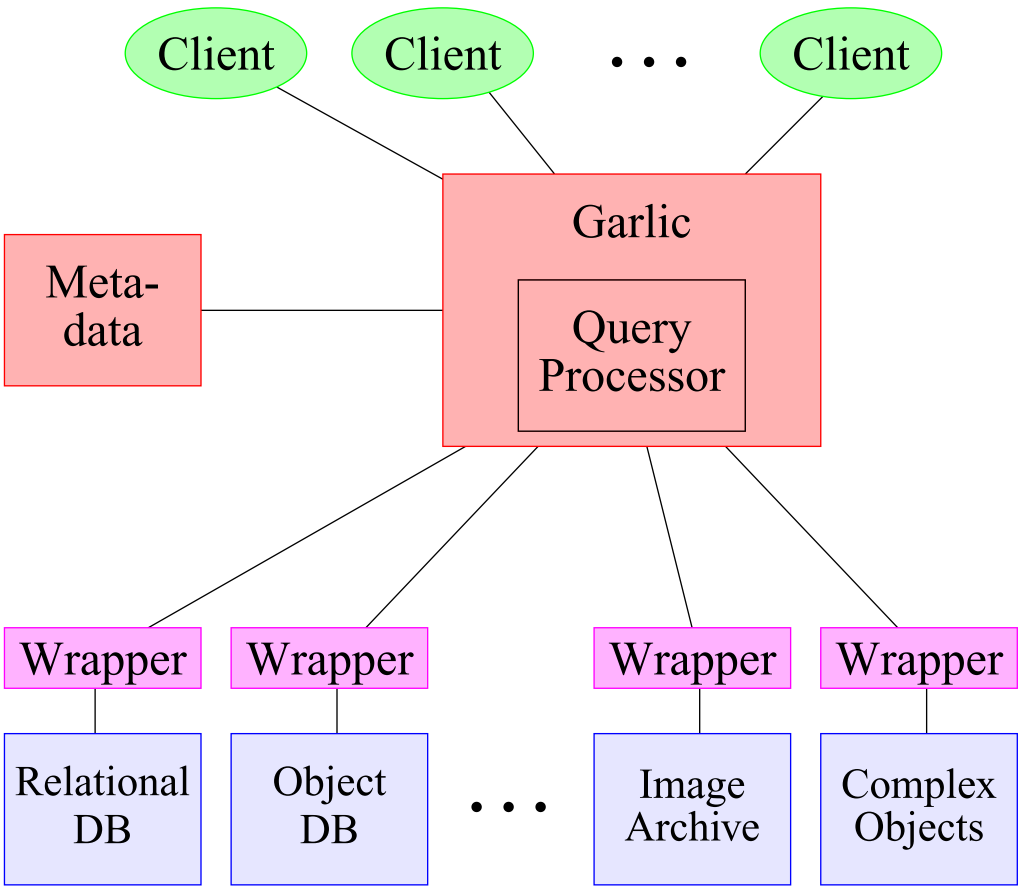
- ❑ Wrapper (allgemein)
- ❑ Garlic (IBM)
- ❑ SQL/MED (Management of External Data)
- ❑ Überblick über OLE (Microsoft)
- ❑ Zusammenfassung

Wrapper

- kapselt die in einer Datenquelle gespeicherten Daten und ‘vermittelt’ zwischen der Datenquelle und der Middleware
- entscheidend hinsichtlich der Bewältigung der Unterschiede (Heterogenität) zwischen den Datenquellen:
  - Wrapper-Architektur
  - Wrapper-Interfaces
- (zu bewältigende) Heterogenität
  - jede Datenquelle hat
    - ihr eigenes Datenmodell und Schema, wobei letzteres sich mit der Zeit ändern kann
    - ihr eigenes API
    - und spezielle Anfragemöglichkeiten
      - Anfragesprache (Spektrum: einfache Scans, Sortierung, einfache Prädikate, komplexe Prädikate, Aggregation, binäre Joins, n-Wege-Joins; evtl. sogar ‘eigentümlich’: spezielle Prädikate nur auf bestimmten Daten erlaubt)
      - Klassen-/Funktionsbibliothek
      - spezifische Programmierschnittstelle

Garlic (1)

- "The wrapper architecture of Garlic ... addresses the challenge of diversity by standardizing how information in data sources is described and accessed, while taking an apporach to query planning in which the wrapper and the middleware dynamically determine the wrapper's role in answering a query"[1]
- Garlic-Architektur



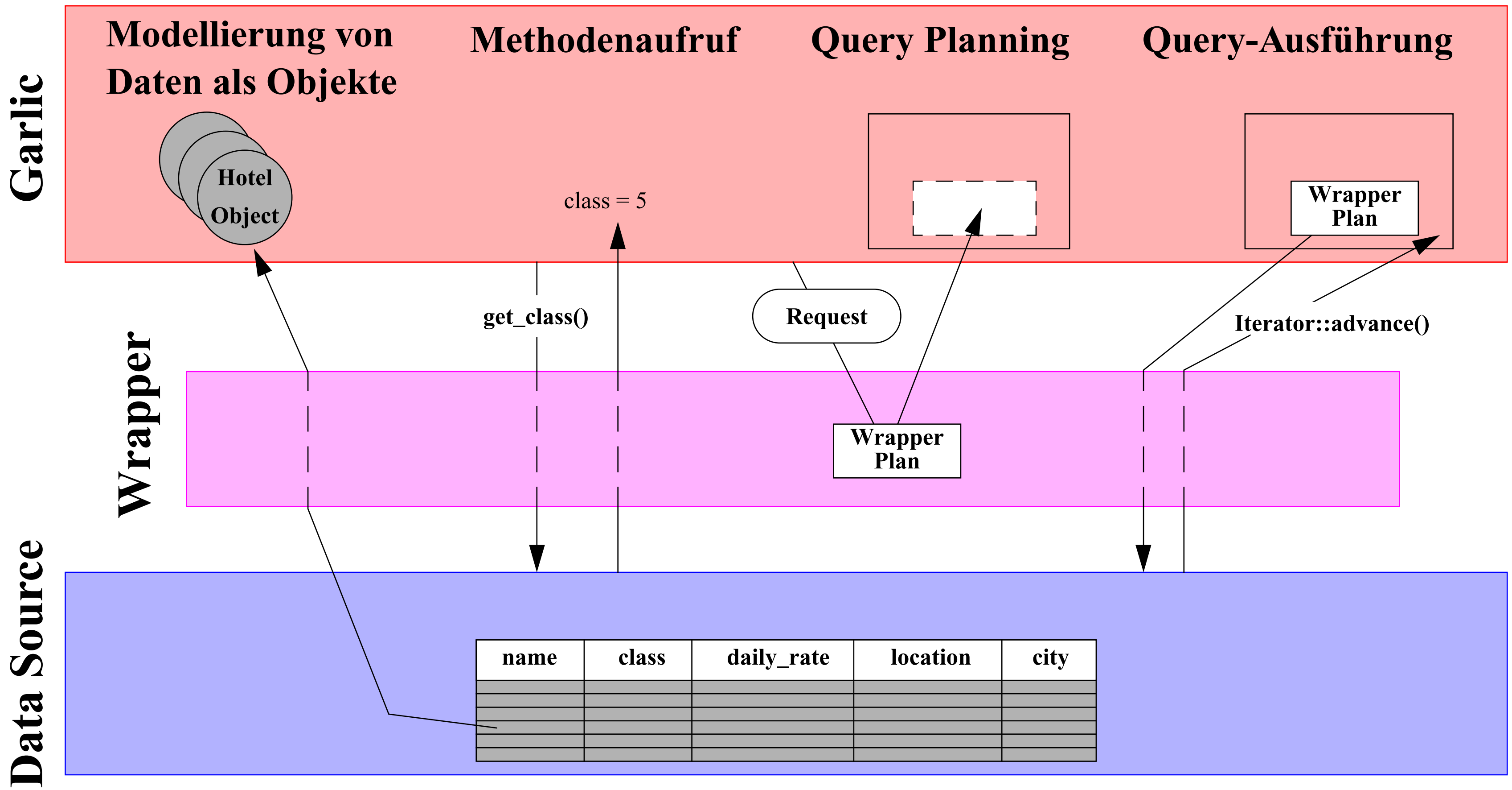
1. M. T. Roth, P. Schwarz: "Don't Scrap It, Wrap It! A Wrapper Architecture for Legacy Data Sources", VLDB'97

Garlic (2)

- ❑ Wrapper-Architektur
  - ➯ Ziele
    - Geringe Start-up-Kosten für die Erstellung eines Wrappers
    - Möglichkeit der Evolution des Wrappers
    - Flexible Architektur; Möglichkeit des ‘Einhängens’ weiterer Datenquellen (werden in Garlic auch *Repositories* genannt) und zugehöriger Wrapper
    - Wrapper sollte der Garlic-Anfrageverarbeitung exakt die Anfragemöglichkeiten verfügbar machen, die die zugehörige Datenquelle unterstützt

Garlic (3)

- Wrapper Services



Garlic (4)

- Modellierung von Daten als Objekte
  - Registrierung
    - Wrapper liefert Garlic Beschreibung der Datenquelle in GDL (Garlic Data Language, Variante von ODMG-ODL)
    - ‘globales Schema’ auf der Garlic-Ebene
  - Garlic-Objekt:
    - Schnittstelle
    - mind. eine Implementierung (evtl. mehrere, höchstens eine pro Datenquelle)
    - Identität: OID besteht aus
      - *IID* (*implementation identifier*)
      - *key* (identifiziert die Ausprägung innerhalb der Datenquelle)
    - Root-Objekte dienen dem Einstieg und können auch über einen externen Namen angesprochen werden

Garlic (5)

- Modellierung von Daten als Objekte (Forts.)
  - Beispiel: Schema einer Reiseagentur

Relational Repository Schema:

```
interface Country {
    attribute string name;
    attribute string airlines_served;
    attribute boolean visa_required;
    attribute Image scene}

interface City {
    attribute string name;
    attribute long population;
    attribute boolean airport;
    attribute Country country;
    attribute Image scene}
```

Web Repository Schema:

```
interface Hotel {
    attribute readonly string name;
    attribute readonly short class;
    attribute readonly double daily_rate;
    attribute readonly string location;
    attribute readonly string city}
```

Image Server Repository Schema:

```
interface Image {
    attribute string file_name;
    double matches (in string file_name);
    void display (in string device_name)}
```

Garlic (6)

❑ Methodenaufruf

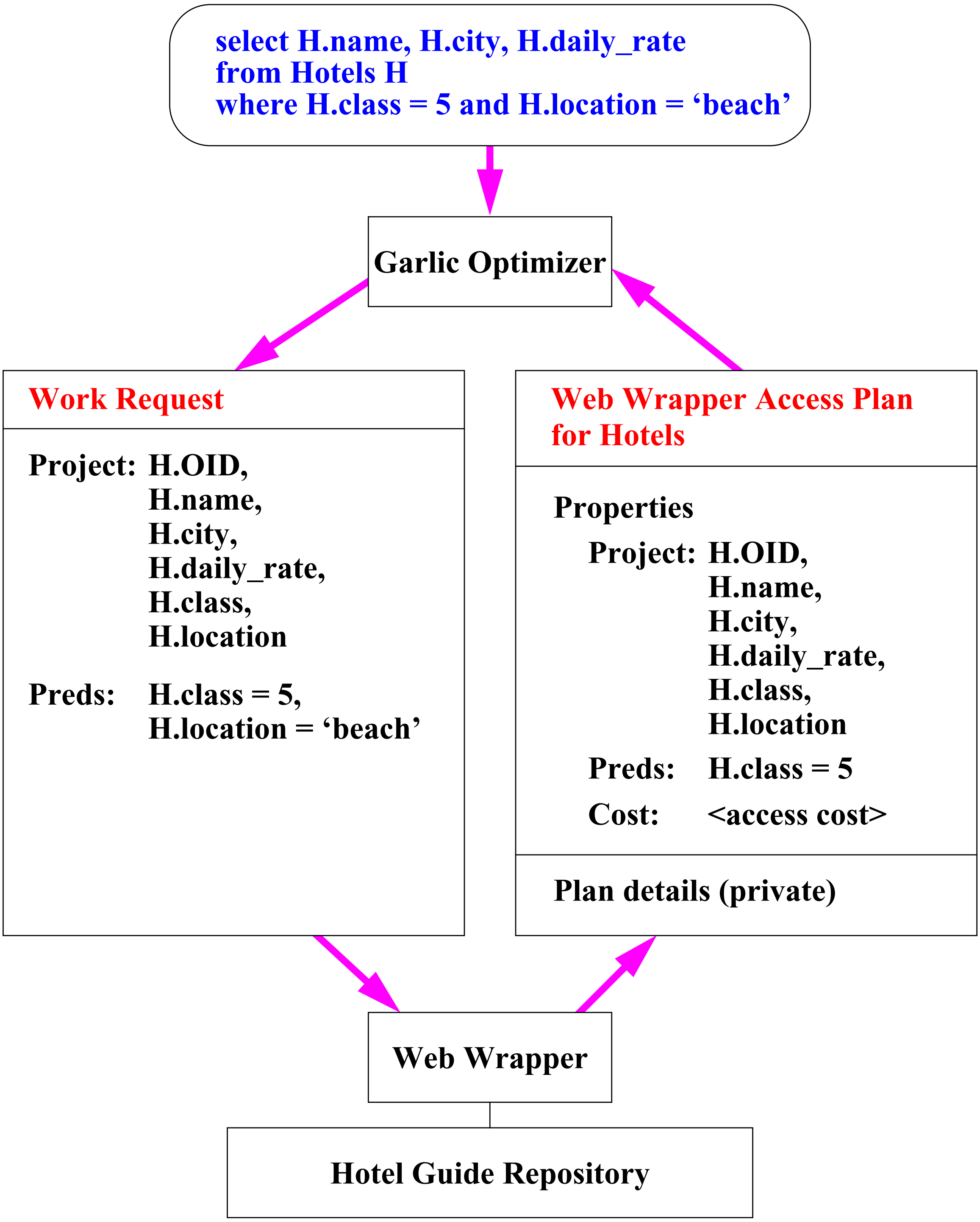
- Methodenaufrufe können durch die Garlic-Ausführungsmaschine bzw. durch eine Garlic-Applikation, die eine Referenz auf ein entsprechendes Repository-Objekt erhalten hat, veranlaßt werden
- Methoden
  - implizit definierte *Get/Set*-Methoden (*accessor methods*)
  - explizit definierte Methoden
- Arten des Methodenaufrufs
  - *stub dispatch*
    - natürlicher Mechanismus für Repositories, deren Schnittstelle als Klassenbibliothek gegeben ist
    - Beispiel: *display* (siehe oben)
      Wrapper stellt Routine zur Verfügung, die aus dem *key*-Feld der OID den Dateinamen des angesprochenen Bildes und aus den von Garlic übergebenen Parametern den Gerätenamen extrahiert; zur Wiedergabe wird dann die zugehörige Funktion der Klassenbibliothek aufgerufen;
  - *generischer Dispatch*
    - Wrapper bietet nur genau einen Methodeneinsprungpunkt
    - schema-unabhängig
    - Beispiel: relationaler Wrapper (siehe oben)
      nur *accessor methods*; jeder Aufruf wird in Anfrage umgesetzt; Methodenname → Attributname, *IID* → Relationenname, Werte → Zuweisung (Set);

Garlic (7)

- Query Planning
  - Grundlegende Idee
    - Wrapper nehmen an der Erstellung des Query-Plans teil
  - Allgemeiner Ablauf:
    - Garlic-Optimierer identifiziert das größtmögliche, ein Repository betreffende Query-Fragment und schickt es zum zugehörigen Wrapper
    - Wrapper liefert einen oder mehrere Pläne zurück, die einen Teil der durch das Query-Fragment angesprochenen Arbeit bzw. die gesamte Arbeit ausführen können
    - Garlic-Optimierer erstellt und beurteilt die möglichen Pläne zur Abarbeitung der gesamten Query; dabei versucht er, für die Teile, die ein Wrapper nicht übernehmen konnte, alternative Operatoren zu finden
  - Wrapper stellt zu diesem Zweck folgende Methoden bereit, die durch *work requests* von Garlic angesprochen werden können:
    - *plan_access()*: generiert *single-collection access plans*
    - *plan_join()*: generiert *multi-way join plans* (Joins können in Anwenderanfragen vorkommen oder von Garlic zur Auflösung von Pfadausdrücken erzeugt werden)
    - *plan_bind()*: generiert speziellen Plan, der als *inner stream* eines *bind joins* benutzt werden kann
  - Ergebnis eines *work requests*:
    - Menge von *plans*
    - jeder *plan* beinhaltet eine Liste von Eigenschaften, die beschreiben, welche Teile des *work requests* durch den *plan* implementiert werden und zu welchen Kosten

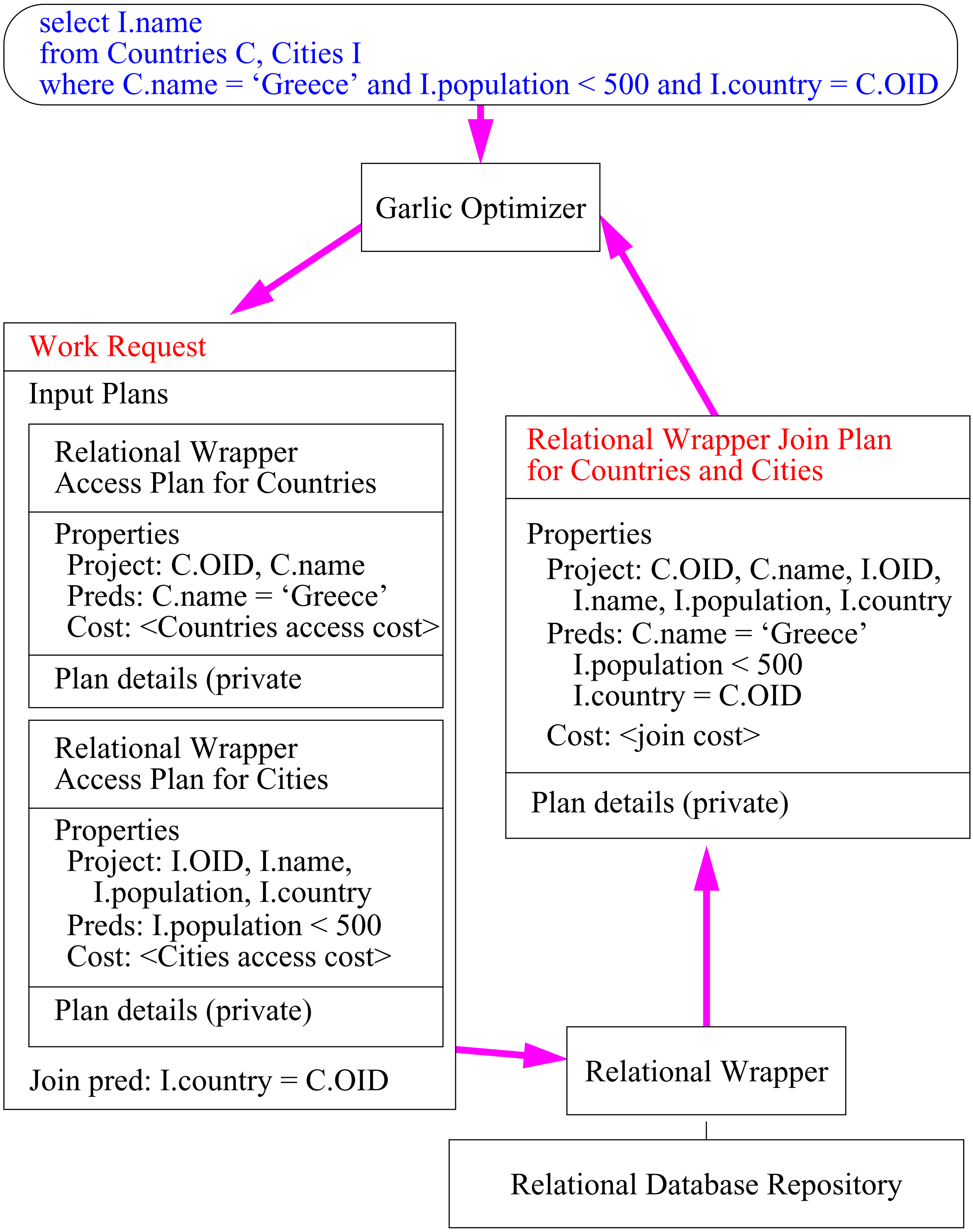
Garlic (8)

- Query Planning (Forts.)
  - *Single Collection Access Plan*



Garlic (9)

- Query Planning (Forts.)
  - *Join Plan*



Garlic (10)

- Query Planning (Forts.)
  - *Bind Plan*

select H.name, H.daily_rate
from Hotels H, Countries C, Cities I
where H.class = 5 and H.location = 'beach' and C.name = 'Greece'
and I.population < 500 and H.city = I.name and I.country = C.OID

Garlic Optimizer

Work Request

Input Plan

Relational Wrapper Join Plan
for Countries and Cities

Properties
Project: C.OID, C.name, I.OID,
I.name, I.population, I.country
Preds: C.name = 'Greece'
I.population < 500
I.country = C.OID
Cost: <join cost>

Plan details (private)

Bind pred: I.name = $BIND_1

Relational Wrapper Bind Plan
for Countries, Cities, and Hotels

Properties
Project: C.OID, C.name, I.OID,
I.name, I.population, I.country
Preds: C.name = 'Greece'
I.population < 500
I.country = C.OID
I.name = $BIND_1
Cost: <bind cost>

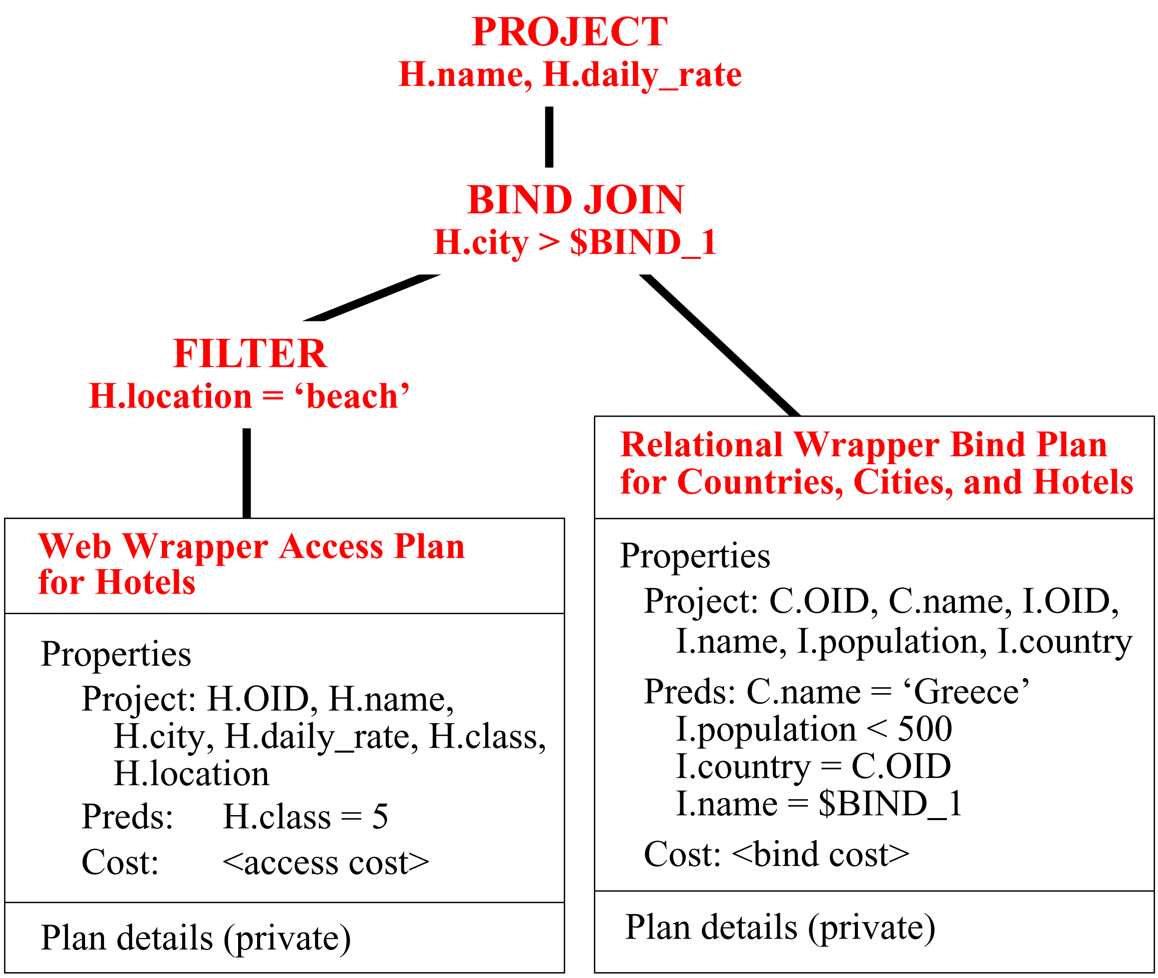
Plan details (private)

Relational Wrapper

Relational Database Repository

Garlic (11)

- Query-Ausführung
  - Synthese der *Wrapper Plans*



  - *Übersetzung* muß unterstützt werden durch entsprechende Wrapper-Methoden
  - *Ausführung* wird ebenfalls durch Wrapper unterstützt (Iterator-Methoden)

Garlic (12)

- Wrapper Packaging
  - Wrapper-Programmierer stellt folgende Wrapper-Komponenten in einem *Wrapper-Package* zur Verfügung:
    - Interface Files
      - GDL-Definitionen
    - Environment Files
      - Kodierung Repository-spezifischer Information
    - Libraries (dynamisch ladbar)
      - Schema-Registrierung
      - Methodenaufruf
      - Anfrageschnittstellen

SQL/MED (1)

- Part 9 von **SQL**:1999: **M**anagement of **E**xternal **D**ata (derzeitiger Status: *Working Draft*)

- zwei wesentliche Teile
  - Datalinks
  - Foreign Data Wrapper / Foreign Data Server

- Datalinks: Konzept
  - ein Datalink(-Wert) ist eine Ausprägung des DATA-LINK-Datentyps
  - ein Datalink referenziert eine Datei (URL), die nicht Teil der SQL-Umgebung ist, sondern durch einen externen File-Manager verwaltet wird
  - Datalink Type Descriptor, Link-Control-Options:
    - link control (NO, FILE)
    - integrity control option (All, SELECTIVE, NONE)
    - read permission option (FS, DB)
    - write permission option (FS, ADMIN, BLOCKED)
    - recovery option (NO, YES)
    - unlink option (RESTORE, DELETE, NONE)
  - Datalinker
    - implementierungsabhängig
    - Menge der Mechanismen, die für durch Datalinks repräsentierte Dateien Integritätskontrolle, Recovery und Zugriffskontrolle durchführen

SQL/MED (2)

- Datalinks, Link-Control-Options (Forts.)
  - NO LINK CONTROL
    - URL-Format des Datalinks
    - keine weitere Kontrolle
  - FILE LINK CONTROL
    - Art der Kontrolle durch die weiteren Optionen bestimmt
  - INTEGRITY ALL
    - referenzierte Dateien können nur über SQL gelöscht oder umbenannt werden
  - INTEGRITY SELECTIVE
    - referenzierte Dateien können mittels File-Manager-Operationen gelöscht oder umbenannt werden, solange kein Datalinker vorhanden ist
  - INTEGRITY NONE
    - referenzierte Dateien können ausschließlich mittels File-Manager-Operationen gelöscht oder umbenannt werden
    - nicht verträglich mit FILE LINK CONTROL
  - READ PERMISSION FS
    - Leserecht für referenzierte Dateien wird durch den File-Manager bestimmt
  - READ PERMISSION DB
    - Leserecht für referenzierte Dateien wird über SQL bestimmt

SQL/MED (3)

- ❑ Datalinks, Link-Control-Options (Forts.)
  - ➪ WRITE PERMISSION FS
    - Schreibrecht für referenzierte Dateien wird durch den File-Manager bestimmt
  - ➪ WRITE PERMISSION ADMIN [NOT] REQUIRING TOKEN FOR UPDATE
    - Schreibrecht für referenzierte Dateien durch SQL bestimmt
  - ➪ WRITE PERMISSION BLOCKED
    - kein Schreibzugriff auf referenzierte Dateien, es sei denn, es existiert implementierungsabhängiger Mechanismus
  - ➪ RECOVERY YES
    - mit SQL-Server koordinierte Recovery (Datalinker-Mechanismus)
  - ➪ RECOVERY NO
    - keine Recovery auf referenzierten Dateien
  - ➪ ON UNLINK RESTORE
    - vor der Herstellung des Links bestehende Rechte (Ownership, Permissions) werden durch den File-Manager bei Auflösung des Links (Unlink) wiederhergestellt
  - ➪ ON UNLINK DELETE
    - Löschung bei Unlink
  - ➪ ON UNLINK NONE
    - keine Auswirkungen auf die Rechte bei Unlink

SQL/MED (4)

- Datalinks, Link-Control-Options (Forts.)
  - Gültige Kombinationen

Integrity	Read permission	Write permission	Recovery	Unlink
ALL	FS	FS	NO	NONE
ALL	FS	BLOCKED	NO	RESTORE
ALL	FS	BLOCKED	YES	RESTORE
ALL	DB	BLOCKED	NO	RESTORE
ALL	DB	BLOCKED	NO	DELETE
ALL	DB	BLOCKED	YES	RESTORE
ALL	DB	BLOCKED	YES	DELETE
ALL	DB	ADMIN	NO	RESTORE
ALL	DB	ADMIN	NO	DELETE
ALL	DB	ADMIN	YES	RESTORE
ALL	DB	ADMIN	YES	DELETE
SELECTIVE	FS	FS	NO	NONE

SQL-MED (5)

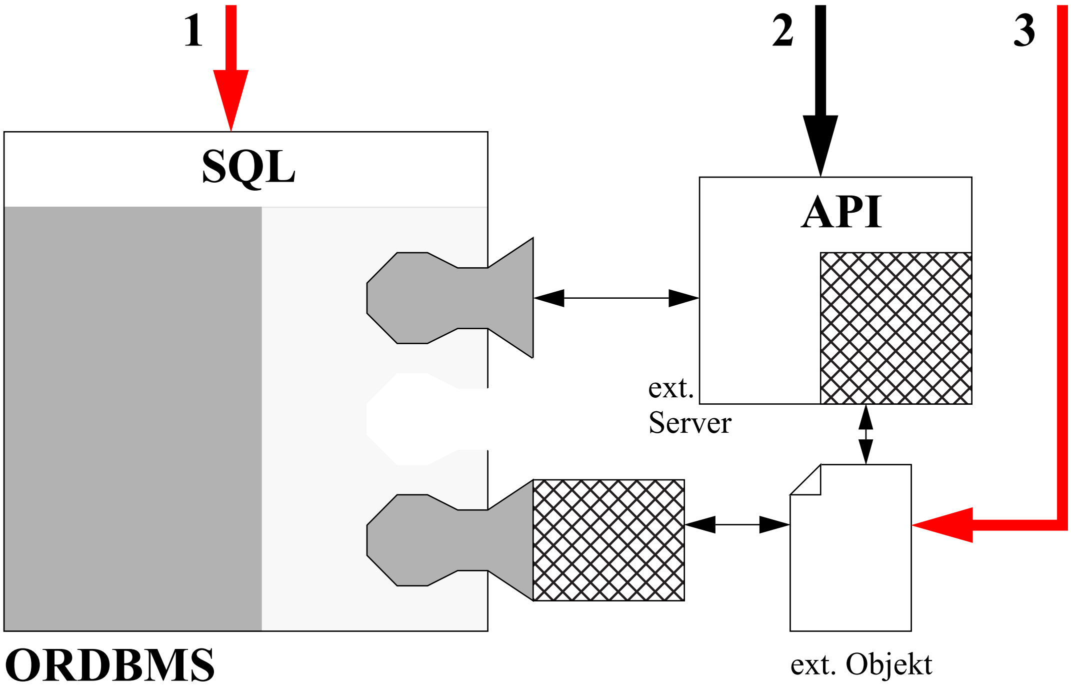
- Neue SQL Funktionen für Datalinks
  - Konstruktor: DLVALUE, ...
  - (Komponenten von) URLs: DLURLCOMPLETE, ...

- SQL Operationen (Beispiele)
  - Einfügen (“Link”)
    - INSERT INTO Movies (Title, Minutes, Movie)
      VALUES (‘My Life’, 126,
      DLVALUE(‘http://my.server.de/movies/mylife.avi’)
  - Selektieren (inkl. URL access token)
    - SELECT Title, **DLURLCOMPLETE**(Movie)
      FROM Movies
      WHERE Title LIKE ‘%Life%’
  - “Unlink/Replace”
    - UPDATE Movies SET Movie =
      DLVALUE(‘http://my.newserver.de/m/life.avi’)
      WHERE Title = ‘My Life’
    - RESTORE oder DELETE für “.../movies/mylife.avi”
  - “Update-in-place”
    - SELECT Title, **DLURLCOMPLETEWRITE**(Movie)
      INTO :t, :url ...
    - Öffnen über URL, Modifizieren ...
    - UPDATE Movies SET Movie = **DLNEWCOPY**(:url, 1)
      WHERE Title = :t

SQL/MED (6)

- Art der Anbindung mittels Datalinks (Forts.)

  - allg. Möglichkeiten der Anbindung externer Daten [1]

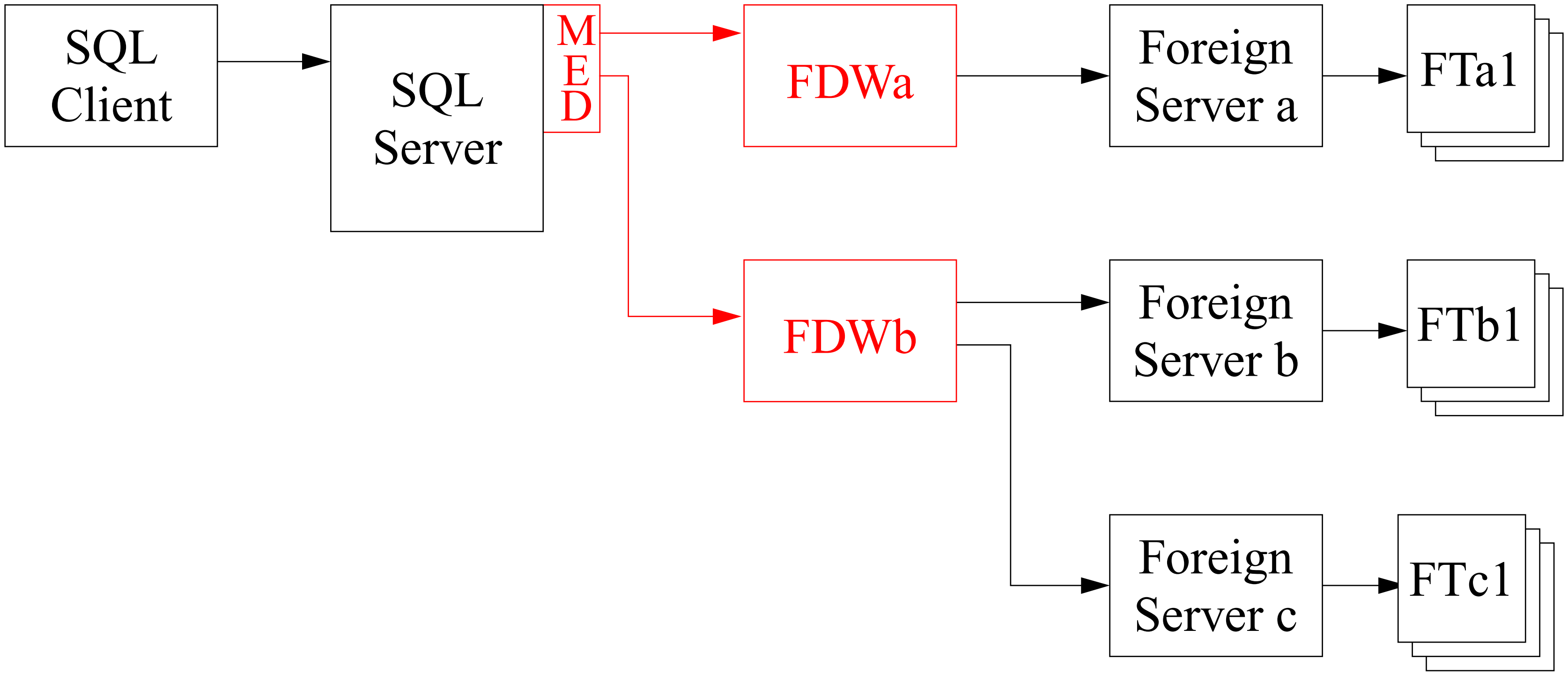


  - Einordnung von Datalinks:
    - externes Objekt = referenzierte Datei
    - **1**: nur Manipulation von URLs (Datalink-Werte)
    - **3**: „überladene“ BS-Zugriffe
      - Rechte- und Integritätskontrolle durch ORDBVS
    - „Einklinken“ von Recovery- und Backup-Mechanismen implementierungsabhängig und durch Standard nicht geregelt

1. Mahnle, Steiert:To a Man with an ORDBMS everythink looks like a Row in a Table, CODAS 2001, Peking.

SQL/MED (7)

- Foreign-Data-Wrapper/Server
  - Konzept basiert auf Garlic-Idee
  - Modell



SQL/MED (8)

❑ Foreign-Data-Server

➪ verwaltet außerhalb der eigentlichen SQL-Umgebung liegende Daten

➪ SQL-Server und SQL-Client nutzen Foreign-Server-Descriptors (Katalogelemente), um mit dem Foreign-Server kommunizieren zu können

➪ Katalog (implementierungsspezifisch):

- SQL-Schemata
- Foreign-Server-Descriptors
- Foreign-Table-Descriptors
- Foreign-Wrapper-Descriptors
- User- and Routine-Mapping-Descriptors

➪ Foreign-Table

- tatsächlich im Foreign-Server (relational) liegende Tabelle oder durch Wrapper-Funktionalität dynamisch zu erzeugende Tabelle

➪ (Interaktions-)Modi

- *Decomposition*
  - Analyse der SQL-Anfrage durch SQL-Server und Kommunikation mit dem Foreign-Data-Wrapper über ***InitRequest***
- *Pass-Trough* (siehe Beschreibung TransmitRequest)

SQL/MED (9)

❑ Foreign-Data-Wrapper (Forts.)

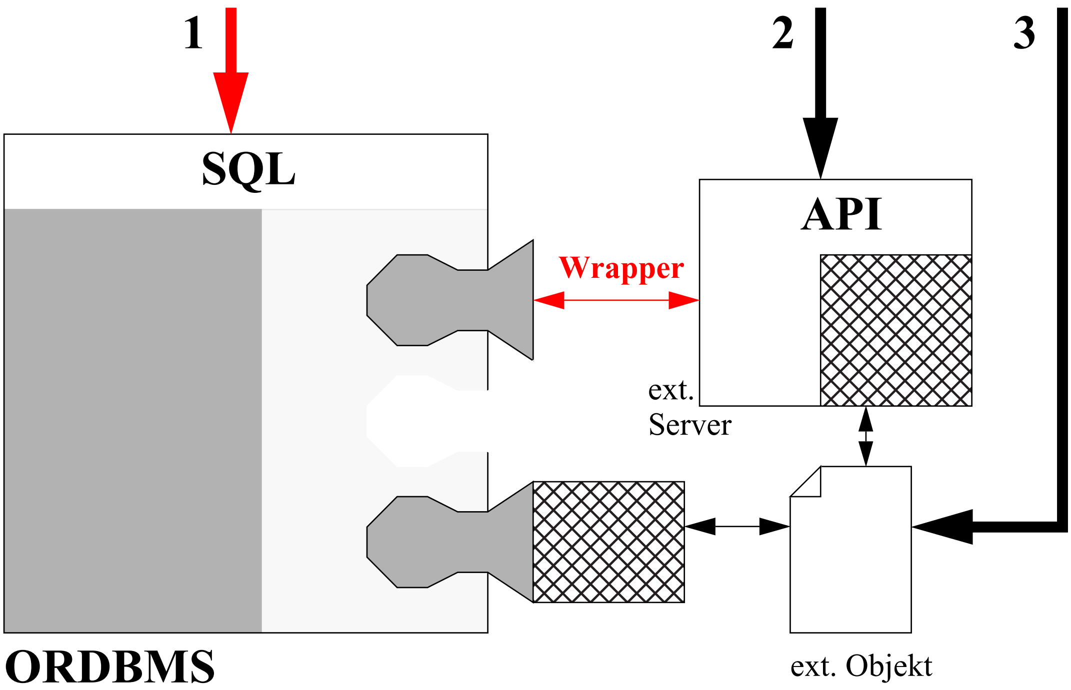
➪ Foreign-Data-Wrapper-Interface

- Handle-Routinen
- Initialisierungsroutinen
  - AllocDescriptor
  - AllocWrapperEnv
  - ConnectServer
  - ***GetOps***: Abfragen von Möglichkeiten des Foreign-Data-Wrappers/Servers, einer (Spalte einer) Foreign-Table
  - **InitRequest**: Initiierung der Vorbereitung einer Anfrage
- Zugriffsroutinen
  - Open
  - ***Iterate***: Übertragung von Tupeln
  - ReOpen
  - Close
  - GetStatistics
  - TransmitRequest: „Durchreichen“ einer Anfrage in der Sprache des Foreign-Data-Servers (Pass-Through-Modus)

SQL/MED (10)

- ❑ Ablauf der Kommunikation zwischen SQL-Server und Foreign-Data-Wrapper folgt dem “request - reply - compensate” Ansatz

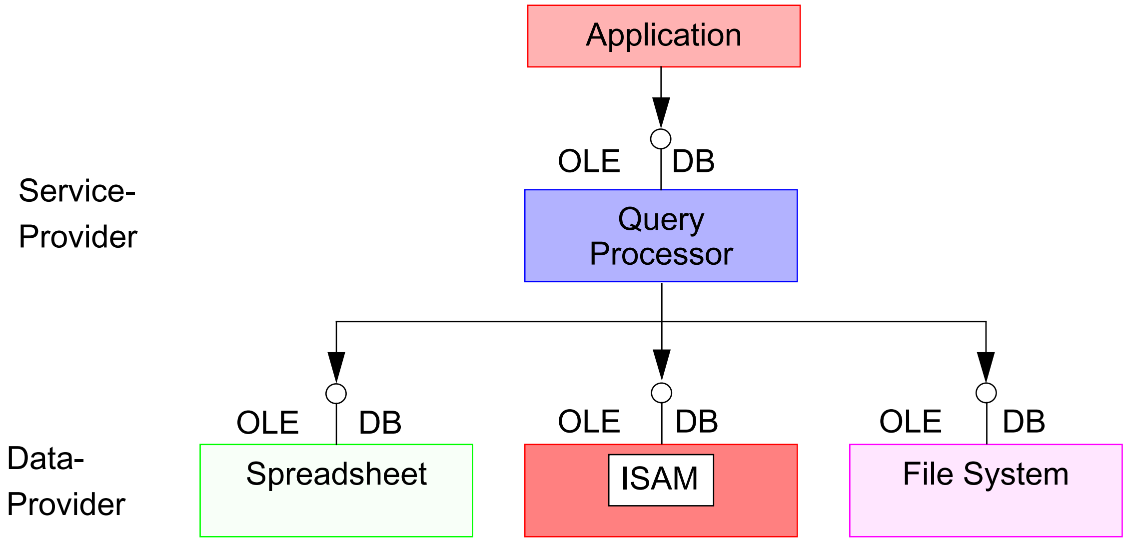
- ❑ Art der Anbindung von Foreign-Data-Wrapper/Server
  - ➪ allg. Möglichkeiten der Anbindung externer Daten



  - ➪ Einordnung von Foreign-Data-Wrapper/Server
    - externe Objekte = Foreign-Tables/Values
    - Manipulation der Daten: **1**, **2**
    - Wrapper implementiert vom SQL-Server erwartete Routinen zur Delegation von Anfrage(teile)n an externen Server

OLE DB[1] von Microsoft

❏ Überblick



➪ Eigenschaften

- ein *Data-Provider* (einfacher Wrapper) kapselt den Zugriff auf D_i und bietet als Abstraktion einen *rowset* als Strom von Datenwerten mit einem Iterator
- Daten, die mehrere *Data-Provider* in Tabellenform verfügbar machen, können durch einen *Service-Provider* zu einer heterogenen Sicht verknüpft werden (Union, Join, Agg. etc.); zum Erstellen eines *Service-Providers* bietet OLE DB spezielle Protokolle; die Middleware-Komponente ist hier wenig generisch
- künftige Versionen sollen komplexere Abstraktionen als flache Tabellen, z. B. objektorientierte und semistrukturierte Daten erlauben

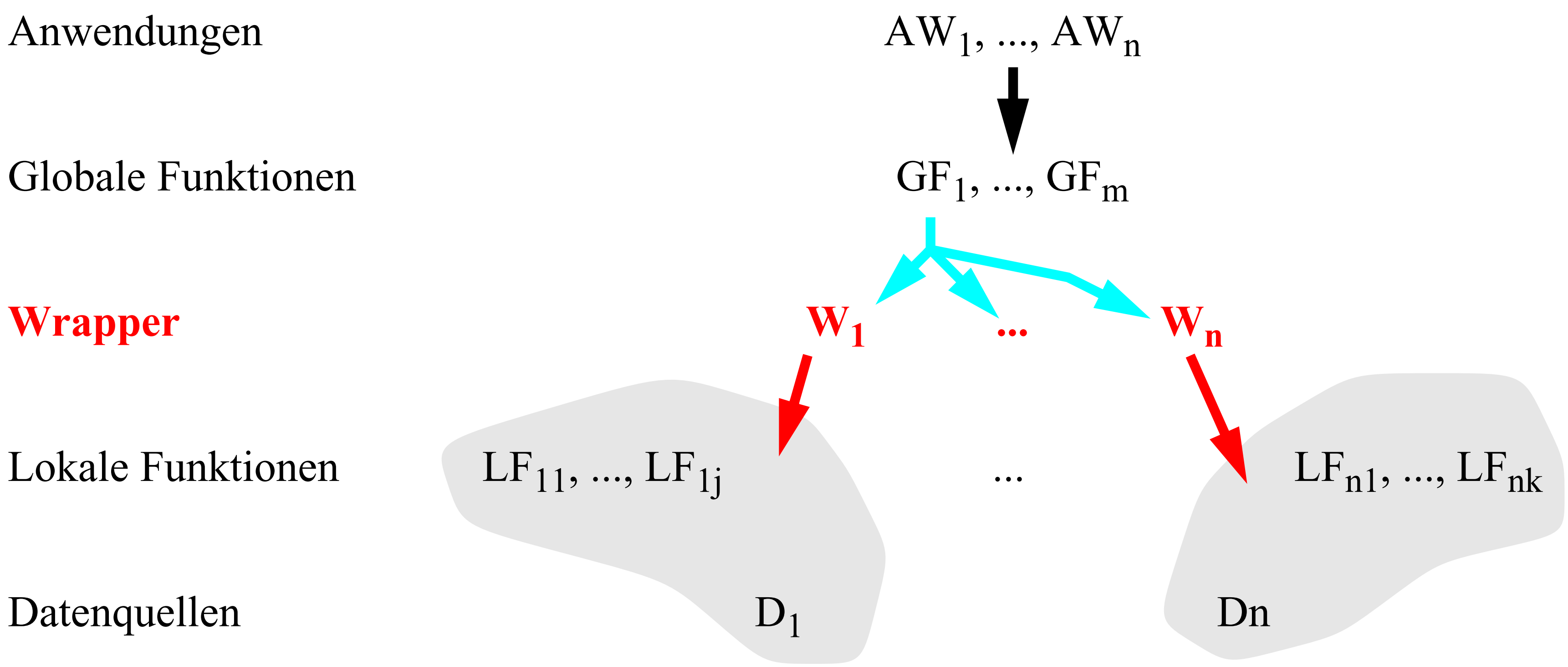
➪ Unterschiede zu Garlic

- Garlic-Anfragefragmente sind in Object-SQL formuliert
- Garlic-Wrapper kann dynamisch seine Beiträge ermitteln

1. J.A. Blakeley: “Universal Data Access with OLE DB”, Proc. IEEE Compcon’97, San Jose, IEEE Computer Society Press, Feb. 1997, pp. 2-7

Zusammenfassung

- Wrapper-Ansätze sind ‘im Inneren’ des Spektrums zwischen Schemaintegration/Anfragetransformation und reiner Funktionsintegration anzuordnen
  - Funktionsintegration



Zusammenfassung

- Wrapper als Mediator zwischen Datenquelle und Middleware
- Beispiel: Garlic (IBM)
  - nahezu jede Art von Datenquelle integrierbar
  - Middleware (Garlic) und Wrapper entscheiden dynamisch, welche Teile der Anfrageverarbeitung durch Wrapper übernommen werden kann
  - spezifische Möglichkeiten der Datenquellen können genutzt werden
- SQL/MED
  - Part 9 des SQL:1999-Standards
  - folgt der Garlic-Idee
  - Foreign-Data-Wrapper/Server
- Spektrum
  - Schemaintegration/Anfragetransformation
  - Wrapper-basierte Ansätze
  - reine Funktionsintegration