Prof. Dr.-Ing. Stefan Deßloch
AG Heterogene Informationssysteme
Geb. 36, Raum 329
Tel. 0631/205 3275
dessloch@informatik.uni-kl.de

Technische Universität Kaiserslautern

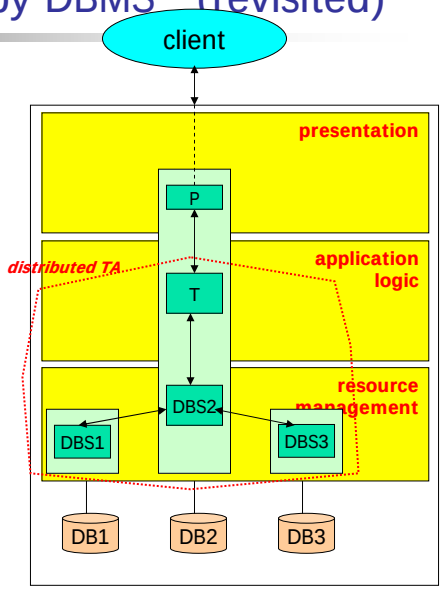
Chapter 9
Virtual Data Integration, Wrappers, and External Data

Outline

- Accessing multiple, distributed data sources in an integrated manner
  - architectures
  - types of transparency achieved
- Wrapper architecture as an infrastructure for overcoming heterogeneity
  - wrapper tasks
  - Garlic
  - SQL/MED
  - OLE-DB
- Accessing and managing external data
  - SQL/MED datalinks

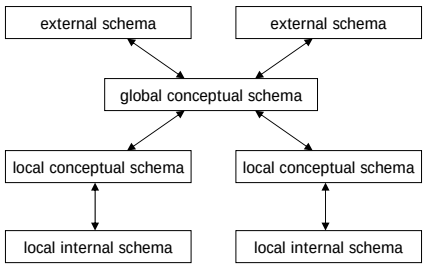
Distribution Controlled by DBMS (revisited)

- Distributed transaction processing
- DB-operation may span across multiple data sources
  - single SELECT statement can access tables in DB1, DB2, DB3
- Virtual integration
  - distributed data storage
  - integrated on demand (query)
- Degrees of transparency
  - location transparency: physical location of data is hidden
  - distribution transparency:
    - fact that data is stored in different data sources is hidden
    - single logical DB and DB-schema for application programmer
- Multiple approaches and architectures possible



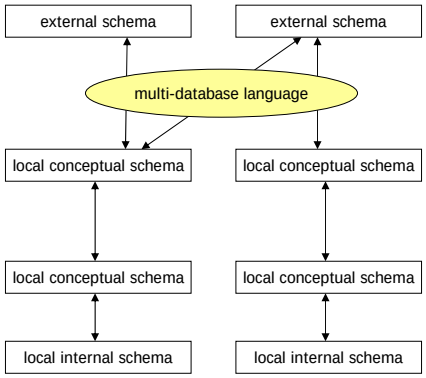
Distributed DBMS

- Data is distributed across multiple systems
  - goal: distribution transparency for applications
- Distribution "by design"
  - distribution is intended, planned
  - participating systems give up autonomy
- 4-layer schema architecture
- Distribution strategies
  - horizontal partitioning
  - vertical partitioning
  - (partial) replication
    - to improve performance
- Tightly coupled
- Heterogeneity is not an issue



Multi-database Systems

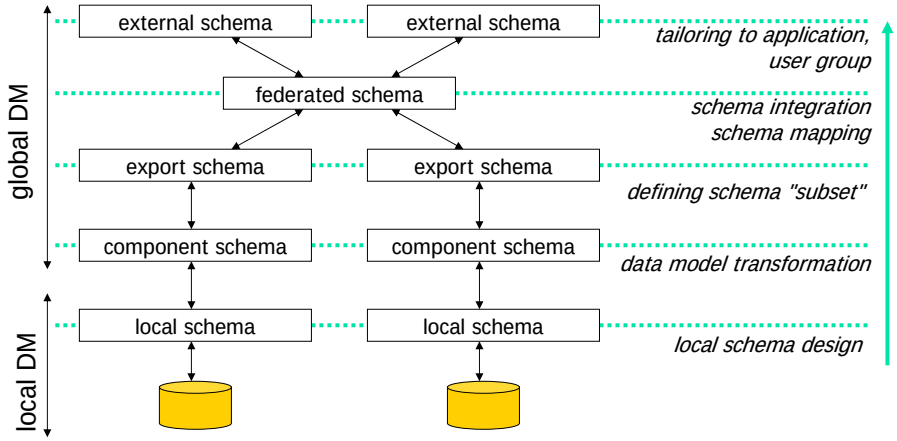
- Loose coupling of systems
  - systems are autonomous
    - schema design autonomy
  - permit external/global applications to access data
- No global schema
  - local export schemas describe available data
  - actual integration needs to be performed by the application
  - only location transparency
- Multi-database language
  - allows access of multiple data bases in a single query
  - directly references export schemas
- Data model heterogeneity needs to be handled either by the local data source or the multi-database language



Federated DBMS

- Based on a global, federated conceptual schema
  - describes integrated view of all participating data sources using the canonical DM
    - global data model (and query language)
  - realizes distribution transparency
  - application can access multiple data sources within the same query
- Distribution is "given", resulting heterogeneity has to be dealt with
  - alternatives for federated schema creation
    - bottom-up: schema integration
    - top-down: schema design, schema mapping
- Preserves high degree of autonomy of participating data sources

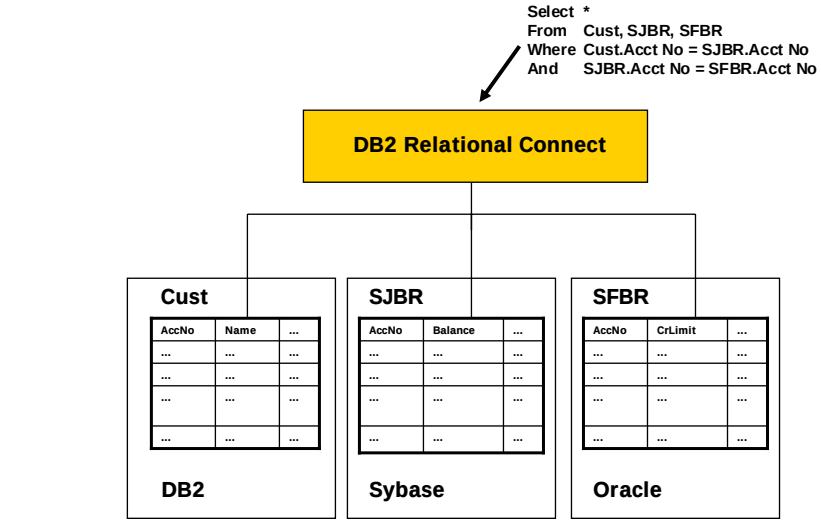
Schema Reference Architecture



More Architecture Components

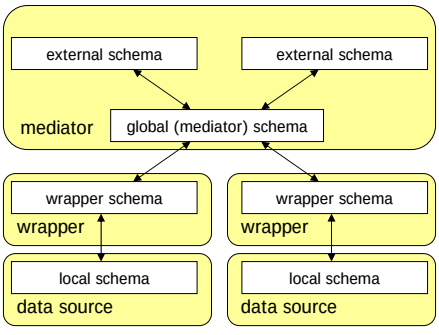
- Local Schema
  - corresponds to the conceptual schema of the local DB
  - based on local data model (e.g., relational DM)
- Component Schema
  - describes the local DB using global (canonical) data model
  - overcome data model heterogeneity
- Export Schema
  - describes subset of local data/schema to be made available for global applications
  - may be under the control of the local system (component system)
    - may result in swapping the export and component schemas in the architecture
- Federated Schema (global schema)
- External Schema
  - corresponds to classic external schema, now for the federated system

Example – Federating Relational Sources



Mediator-based Information Systems

- Generalization of previous architectures based on
  - wrappers for accessing data sources
  - mediators that access one or more wrappers and provide useful services
    - search/query
    - data transformation
    - providing meta-data
    - data-level integration
    - ...
- Data sources remain autonomous
- Global mediator schema to achieve distribution transparency
- Architecture variations
  - nesting of mediators
  - single wrapper for multiple sources (of the same type)
  - applications accessing wrappers directly

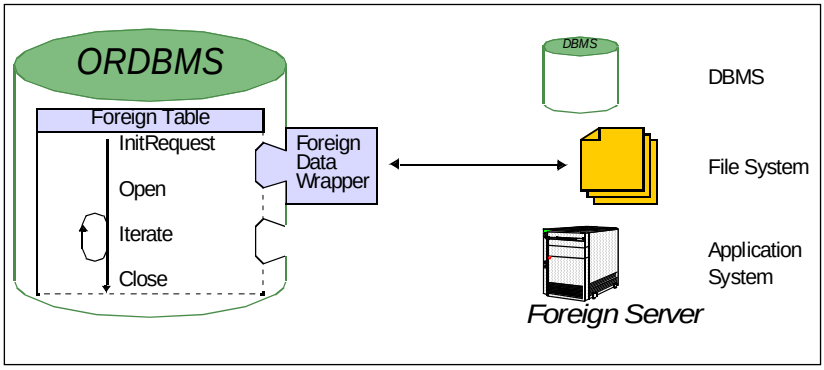


Wrapper Tasks

- Encapsulate a data source and provide uniform access to data
- Provide infrastructure for overcoming heterogeneity among data sources:
  - wrapper architecture
  - wrapper interfaces
- Help overcome heterogeneity of data sources
  - data model, schematic heterogeneity
  - data access API
  - query language and capabilities
    - query language and expressiveness (simple scan, sort, simple predicates, complex predicates aggregation, binary joins, n-way joins, ...)
    - class/function libraries
    - proprietary query APIs
- Support global query evaluation and optimization
  - provide information about ability to process parts of a query
    - cost information
- ➔ Useful infrastructure for Federated DBMS

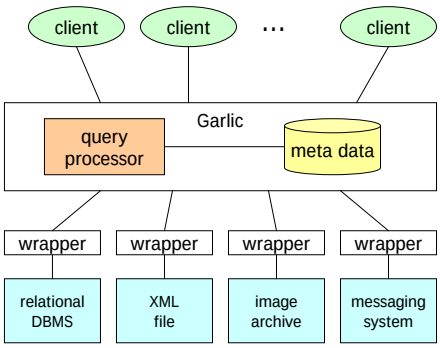
SQL – Management of External Data (MED)

- 'Foreign Data Wrapper' in 'SQL/MED'



Garlic

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

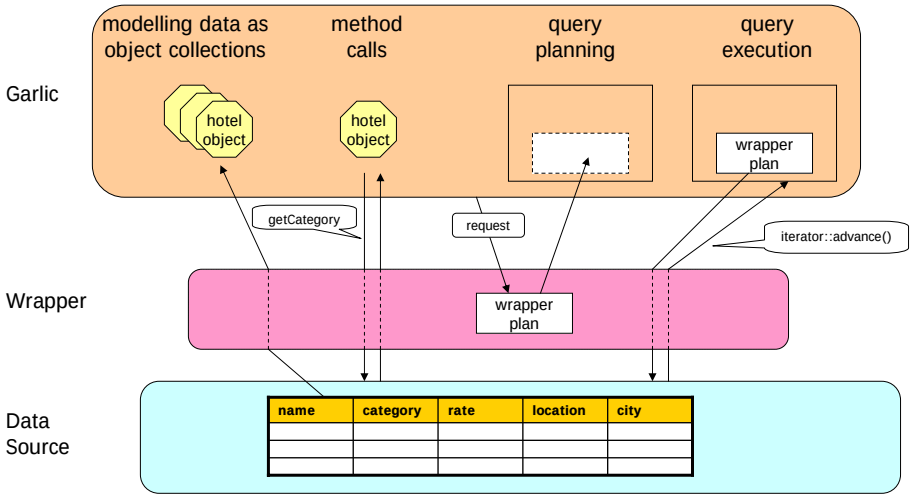


M. T. Roth, P. Schwarz: "Don't Scrap It, Wrap It! A Wrapper Architecture for Legacy Data Sources", VLDB'97

Wrapper Architecture

- Garlic and wrappers cooperate for query processing
  - wrapper provides information about its processing capabilities
  - Garlic query engine compensates for (potential) lack of wrapper query functionality
    - function compensation
- Extensibility
  - add new data sources (accessed using existing wrappers)
  - add new wrappers for supporting new types of data sources
- Wrapper evolution
  - start with simple wrappers (equivalent of a table/collection scan)
    - low cost
  - expand query processing capabilities of the wrapper until it provides full support of the data source functionality

Wrapper-Services



Modeling Data as Object Collections

- Registration
  - wrapper supplies description of data source in GDL (Garlic Data Language, derived from ODMG-ODL)
  - 'global schema' at the garlic level
- Garlic object
  - interface
  - at least one implementation (multiple are possible, but only one per data source
  - identity: OID consists of
    - IID (implementation identifier)
    - key (identifies instance within a data source)
  - root objects (collections) serve as entry into data source, can be referenced using external names

Example: Travel Agency Schema

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 category;
    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)}
```


Method Calls

- Method can be called by Garlic query execution engine or by the application, based on an object reference
- Methods
  - implicitly defined get/set-methods (accessor methods)
  - explicitly defined methods
- Invocation mechanisms
  - stub dispatch
    - natural if data source provides object class libraries
    - example: *display* (see previous charts)
      wrapper provides routine that extracts file name from OID, receives device name as parameter, calls class library for display operation
  - generic dispatch
    - wrapper provides one entry point
    - schema-independent
    - example: relational wrapper (see previous charts)
      access methods only; each call is translated into a query:
      - method name -> attribute
      - IID -> relation name
      - value -> assignment value (SET)

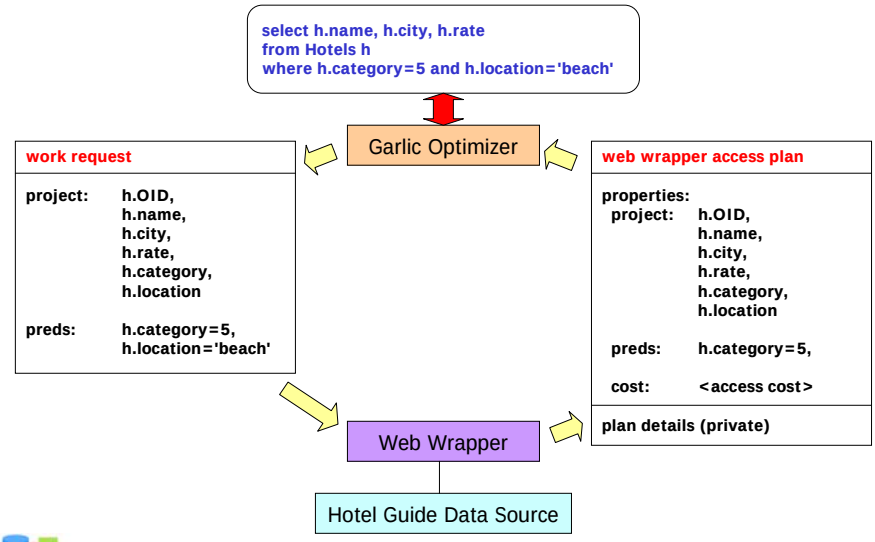
Query Planning

- Fundamental Idea: wrappers participate in query planning process
- Query planning steps
  - Garlic optimizer identifies for each data source the largest possible query fragment that does not reference other data sources, sends it to the wrapper
  - wrapper returns one or more query plans that can be used to process the full query fragment or parts of the query fragment
    - providing all objects in a collection is minimal requirement
  - optimizer generates alternative plans, estimates execution costs, provides for compensation of fragments not supported by the wrapper

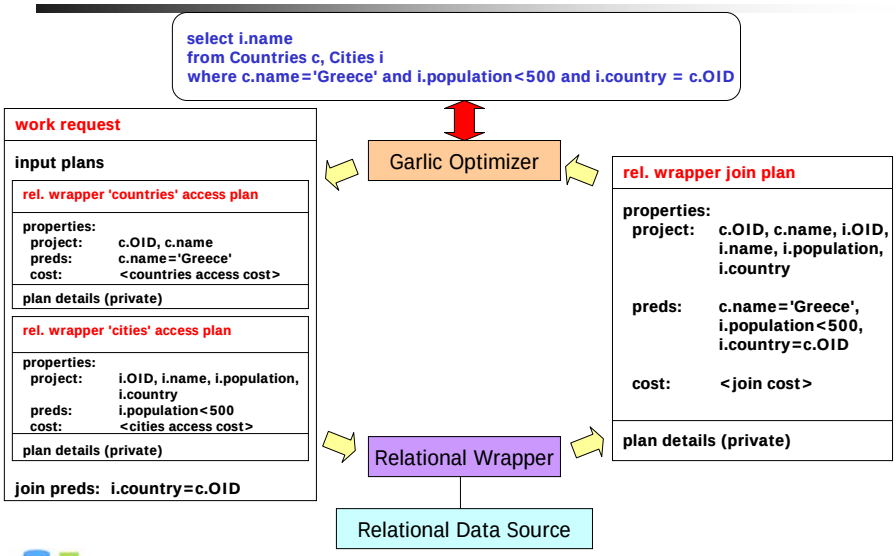
Query Planning (continued)

- Wrapper provides the following methods to be used by Garlic *work requests*:
  - *plan_access()*: generates *single-collection access plans*
  - *plan_join()*: generates *multi-way join plans* (joins may occur in application queries or in the context of resolving path expressions)
    - tables to be joined all reside in the same data source
  - *plan_bind()*: generates special plan, which can be used to process the *inner stream* of a *bind join*
- Result of a *work request*:
  - sets of *plans*
  - each *plan* contains a list of properties that describe which parts of the work request are implemented by the plan and what the costs are

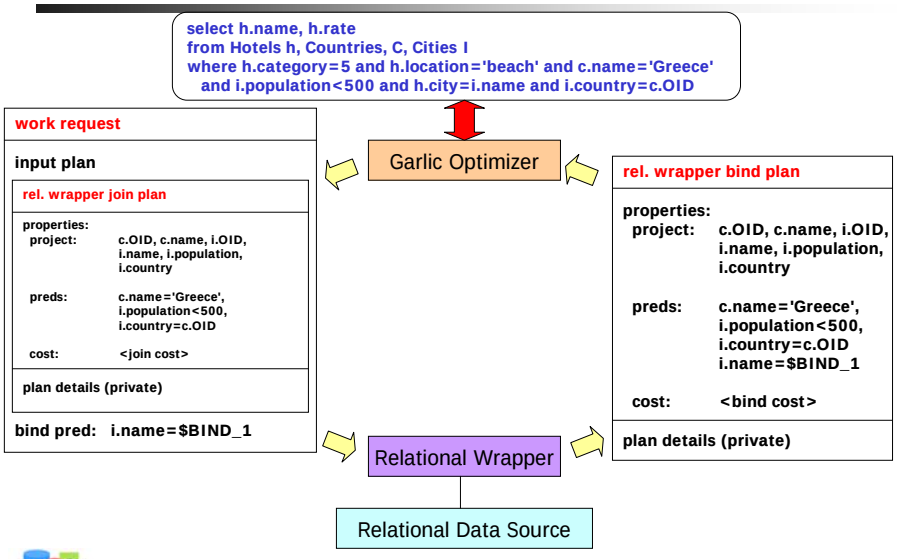
Single Collection Access Plan



Join Plan

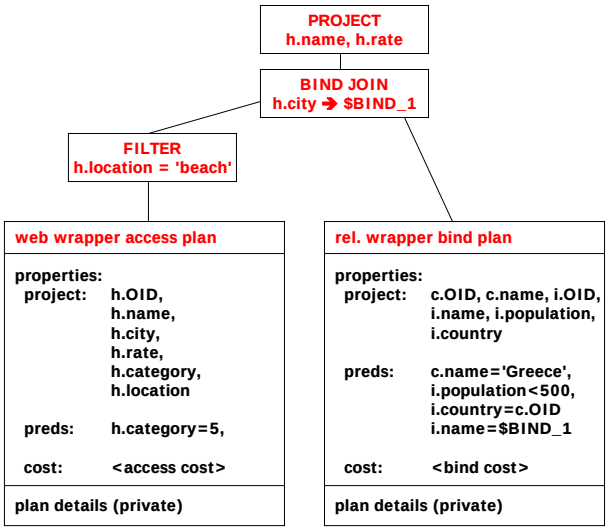


Bind Plan



Wrapper Plan Synthesis

- Plan generation needs to be supported by wrapper methods
- Plan execution has to be supported by wrapper as well (Iterator methods)



Wrapper Packaging

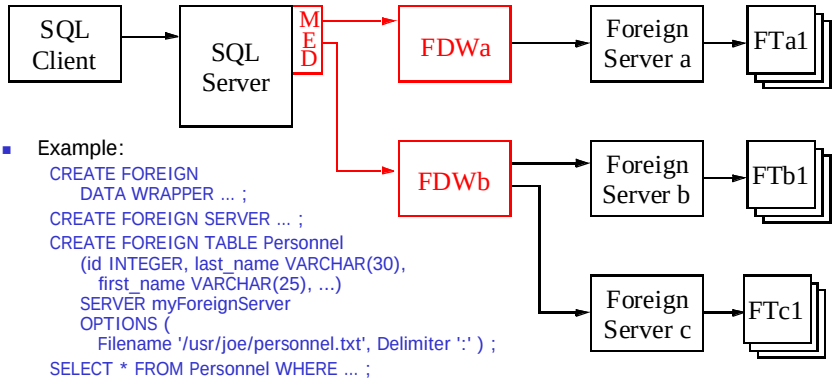
- Wrapper program provides the following wrapper components in a package:
  - interface files
    - GDL definitions
  - environment files
    - support for data-source-specific information
  - libraries
    - schema registration
    - method calls
    - query processing interfaces

SQL/MED

- Part 9 of **SQL**:1999: **M**anagement of **E**xternal **D**ata
  - extended in SQL:2003
- Two major parts
  - Datalinks
  - Foreign Data Wrapper / Foreign Data Server

Foreign Data Wrapper/Server

- Concept based on Garlic idea
  - data provided as tables instead of object collections
- Model:



- Example:

```
CREATE FOREIGN
    DATA WRAPPER ... ;
CREATE FOREIGN SERVER ... ;
CREATE FOREIGN TABLE Personnel
    (id INTEGER, last_name VARCHAR(30),
      first_name VARCHAR(25), ...)
    SERVER myForeignServer
    OPTIONS (
      Filename '/usr/joe/personnel.txt', Delimiter ':' ) ;
SELECT * FROM Personnel WHERE ... ;
```


Foreign Data Server

- Manages data stored outside the SQL server
- SQL server and SQL client use foreign server descriptors (catalog elements) to communicate with foreign servers
- Catalog (implementation-specific):
  - SQL schemas
  - Foreign server descriptors
  - Foreign table descriptors
  - Foreign wrapper descriptors
- Foreign table
  - stored in a (relational) foreign server or dynamically generated by foreign wrapper capabilities
- Modes of interaction
  - *Decomposition*
    - SQL query is analyzed by SQL server, communicating with foreign data wrapper using *InitRequest*
  - *Pass-Through* (see discussion of *TransmitRequest*)

Foreign Data Wrapper Interface

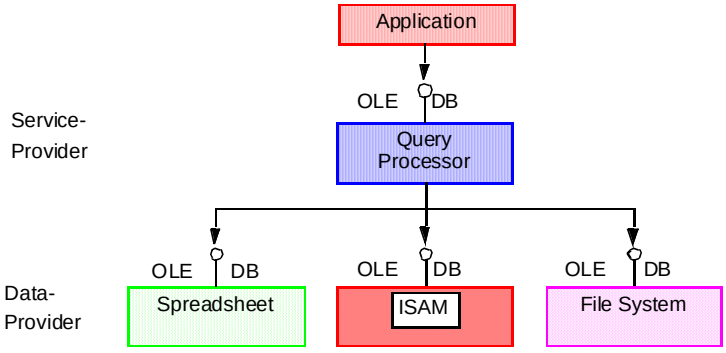
- Handle routines
- Initialization routines
  - AllocDescriptor
  - AllocWrapperEnv
  - ConnectServer
  - GetOps: request meta data about
    - foreign data wrapper/server capabilities
    - foreign table (columns)
  - InitRequest: initializes processing of a request (query)
- Access routines
  - Open
  - Iterate: for delivering foreign data to SQL server
  - ReOpen
  - Close
  - GetStatistics
  - TransmitRequest: „pass-through“ of a query/request using the proprietary language of the foreign server

Security, Updates, and Transactions

- User Mapping
  - defines mapping of SQL server user-ids to corresponding concept of a foreign server
  - example:
    - CREATE USER MAPPING FOR dessloch
      SERVER myforeignserver
      OPTIONS
      (user_id 'SD',
      user_pw 'secret')
- Updates, transactions on external data
  - not supported in SQL/MED
    - goal for future version of the standard
  - provided as product extensions
    - usually, updates on non-relational data sources are not supported
      - distributed TAs are useful, read-only optimization can be used for foreign data source
    - updates on relational data sources
      - pass-through
      - transparent
      - distributed TAs supported

Microsoft OLE-DB

- Overview

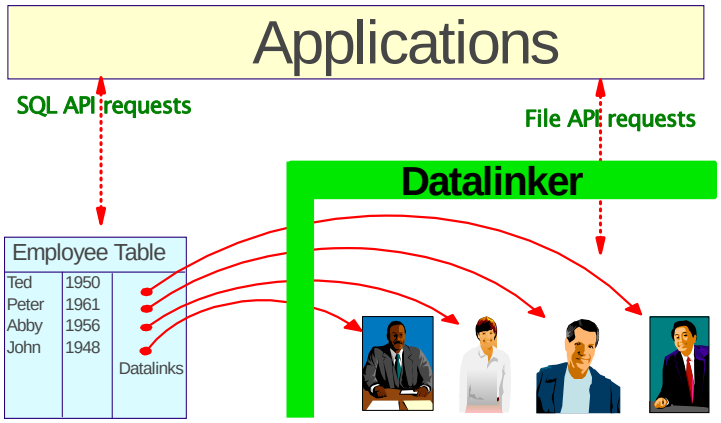


J.A. Blakeley: "Universal Data Access with OLE DB",
Proc. IEEE Compcon'97, San Jose, IEEE Computer Society Press, Feb. 1997

Concepts

- Data provider
  - simple wrapper
  - encapsulates data source access
  - provides a rowset abstraction to allow iteration over a stream of data values
- Service provider
  - can provide view over heterogeneous sources by combining rowsets from different providers
    - union, join, aggregation, ...
  - special OLE-DB protocols for service provider implementation
- Recent enhancements for going beyond "flat" rowsets
- Garlic vs. OLE-DB service provider
  - Garlic queries use Object-SQL
  - Garlic wrapper and query processor interact dynamically for determining query plan

Managing External Data: Datalinks



DataLinks in SQL/MED

- Goal
  - preserve external storage, manipulation of files
  - synchronize integrity control, recovery, and access control of files and associated SQL data
- Concepts
  - datalink is an instance of the DATALINK data type
    - references a file (URL) that is not stored by the SQL server, but maintained by an external file manager
  - datalink options (per column)
    - define the amount of management and control the SQL server has over the datalink values of a column
      - integrity, read/write access, recovery
    - specifies the semantics of link/unlink behavior
  - datalinker
    - implementation-dependent
    - implements a number of mechanisms for guaranteeing datalink properties such as integrity control, recovery, access control

Functions and Operations

- New SQL functions for datalinks
  - constructor: DLVALUE, ...
  - (components of) URLs: DLURLCOMPLETE, …
- SQL statements (examples)
  - insert (“link”)

```
INSERT INTO Movies (Title, Minutes, Movie)
    VALUES ('My Life', 126,
    DLVALUE('http://my.server.de/movies/mylife.avi'))
```

  - select (incl. URL access token)

```
SELECT Title, DLURLCOMPLETE(Movie)
    FROM Movies
    WHERE Title LIKE '%Life%'
```


Data Link Options

- Link control (NO, FILE)
  - NO LINK CONTROL
    - URL-Format of datalink
    - no further control, file is not "linked"
  - FILE LINK CONTROL
    - file is "linked", file has to exist!
    - level of control can be specified using further options
- Integrity control option (ALL, SELECTIVE, NONE)
  - INTEGRITY ALL
    - linked files cannot be deleted or renamed
  - INTEGRITY SELECTIVE
    - linked files can only be deleted or modified using file manager operations, if no datalinker is installed
  - INTEGRITY NONE
    - referenced files can be deleted or modified using file manager operations
      - not compatible with FILE LINK CONTROL

Data Link Options (continued)

- Read permission option (FS, DB)
  - READ PERMISSION FS
    - read access is determined by file manager
  - READ PERMISSION DB
    - read access is controlled by SQL server, based on access privileges to the datalink value
    - involves read access tokens
      - encoded into the URL by the SQL server
      - verified by external file manager/data linker
- Write permission option (FS, ADMIN, BLOCKED)
  - WRITE PERMISSION FS
    - write access controlled by file manager
  - WRITE PERMISSION BLOCKED
    - linked files cannot be modified
  - WRITE PERMISSION ADMIN [NOT] REQUIRING TOKEN FOR UPDATE
    - write access governed by SQL server (and datalinker)
      - requires READ PERMISSION DB
    - involves write access token for modifying file content
      - may have to be presented to the SQL server again

Functions and Operations (continued)

- "Update-in-place"

```
SELECT Title, DLURLCOMPLETEWRITE(Movie)
    INTO :t, :url ...

open using URL, modify ...

UPDATE Movies SET Movie = DLNEWCOPY(:url, 1)
    WHERE Title = :t
```

- DLNEWCOPY
  - indicates to the SQL server that the file content has changed and should be managed appropriately
  - alternative: DLPREVIOUSCOPY – file content may have changed, but the application is not interested in keeping the changes, original file is restored

Data Link Options (continued)

- RECOVERY YES/NO
  - indicates whether SQL server coordinates recovery (jointly with datalinker) or not
- Unlink option (RESTORE, DELETE, NONE)
  - ON UNLINK RESTORE
    - original properties (ownership, permissions) restored as well
  - ON UNLINK DELETE
    - file is deleted when unlinked
  - ON UNLINK NONE
    - ownership and permissions are not restored
- SQL statement (example)
  - "Unlink/Replace"

```
UPDATE Movies SET Movie =
   DLVALUE('http://my.newserver.de/mylife.avi')
   WHERE Title = 'My Life'
```

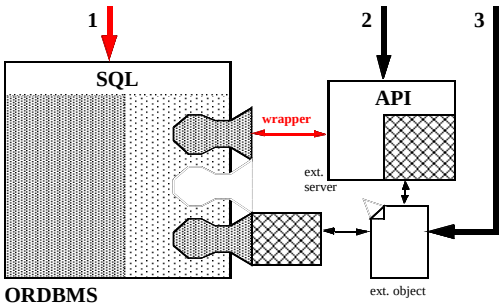
RESTORE or DELETE for ".../movies/mylife.avi"

Valid Combinations

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

Comparing Wrappers and Data Links

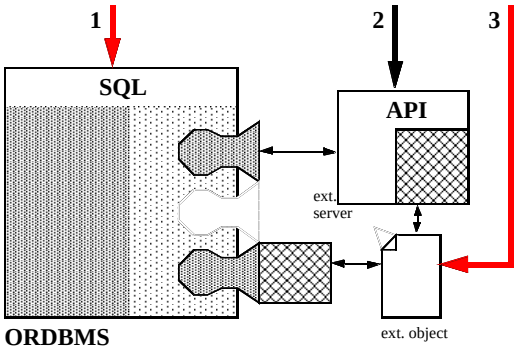
- Choices for accessing/manipulating external data



- For foreign data wrapper/server
  - external object = foreign tables/values
  - manipulation of data: 1
  - wrapper implements routines needed to interact with SQL server for delegation of query fragments to external server

Comparing Wrappers and Data Links (cont.)

- For datalinks:
  - external object = referenced file
  - 1: manipulation of URLs (datalink values), obtaining permissions (token)
  - 3: "overloaded" file system access
    - access and integrity control through ORDBMS
  - Synchronization of recovery- and backup-mechanisms implementation-dependent, not defined by standard



Summary

- Architectures for virtual data integration
  - distributed DBMS, federated DBMS, mediator-based systems
    - based on a global schema, can support location and distribution transparency
  - multi-database systems
    - no global schema, only support location transparency
- Wrappers as important infrastructure
  - Advantages
    - provide a common interface for integration middleware to interact with arbitrary data sources
    - overcomes heterogeneity regarding data model, API
  - Garlic (IBM)
    - almost any data source can be integrated
    - global query optimization
      - middleware (Garlic) and wrapper decide dynamically which query fragments are processed by the wrapper
      - specific capabilities of data sources can be utilized
      - function compensation by Garlic for query capabilities not supported by a wrapper

Summary (cont.)

- SQL/MED – Management of External Data
  - Foreign-Data-Wrapper/Server/Table
    - provides standardized support for extending SQL engine to access external data sources
    - follows the Garlic idea
    - Limitations: no standardized update operations, no transactional support
  - Datalinks
    - extends SQL engine to manage external files
      - SQL-based access to file URL, based on datalink data type
      - file-system-based access to file content
    - alternative to SQL large object types
      - LOBs provide SQL-based storage and access of content
      - requires file content migration and application changes for existing content
    - support referential integrity for external files
    - transactional integration
    - integrate recovery and access control of files and SQL data