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

TECHNISCHE UNIVERSITÄT
KAISERSLAUTERN

Enterprise Information Systems
Chapter 1 - Motivation

EIS – Main Focus Is On Integration

- Data/Information Integration
  - integrated access to (heterogeneous) data originating from multiple sources
    - queries range over date from multiple DBs!
  - virtual integration: integrate on access/query (e.g., federated DBMS)
  - materialized integration: extract, transform, load data into a single materialized data warehouse in advance (e.g., data replication, data warehousing)
  - needs a strong foundation to overcome multiple kinds of heterogeneity
- Enterprise Application Integration
  - integration of (heterogeneous, coarse-grained) applications within an enterprise (vs. development of new application)
  - integration across different middleware platforms
- Business-to-business Integration
  - support interactions, integration of business processes among trading partners, across company boundaries
  - foundation for e-business, e-commerce

Integration Challenges

- Goal of Integration:
  Provide a homogeneous, integrated view on multiple, distributed, autonomous and heterogeneous systems, components, or data sources.
- Three fundamental challenges:
  - Distribution
  - Autonomy
  - Heterogeneity
- Orthogonal, but interrelated

Let's look at the above challenges in the scope of data/information integration!

Distribution

- Physical distribution
  - Data located on (geographically) separated systems
  - Challenges:
    - Addressing data across the globe (URLs)
    - Accessing data in different schemas (Multi-database languages, federated database systems)
    - Optimizing distributed queries (no topic of this lecture)
- Logical distribution
  - Several possible storage locations for a given data item
  - Caused by (partial) redundancy due to overlapping intension of schema elements
  - Challenges:
    - Maintaining consistency among redundant data
    - Provide metadata to enable data localization
    - Detect and resolve duplicates
    - Detect and resolve data inconsistencies and conflicts

} Data Cleaning

- Physical and logical distribution are orthogonal:
  - Data can be logically distributed and physically on the same system (and vice versa)

Autonomy

- Design Autonomy
  - Administrators of data sources can freely decide in which way they model data
  - Data model, formats, units, …
  - Leads to heterogeneity among sources
- Interface Autonomy
  - Freedom to decide how technical access is provided
  - Protocols (HTTP, JDBC, SOAP, …), supported query languages (SQL, XQuery, …)
- Access Autonomy
  - Freedom to decide *whom* to allow access to *what* data
  - Mode of Authentication (Certificates, Username/Password)
  - Authorization (boolean, R/W, Access Control Lists, …)
- Judicial Autonomy
  - Freedom to prohibit integration of data by others
  - Intellectual property (IP) issues

⇨ Autonomy is the major cause of integration problems

Heterogeneity

- Translated from [LeNa07]:
  "Two information systems that do not provide the exact same methods, models and structures to access their data are called heterogeneous."
- Causes for heterogeneity among IS:
  - Specific requirements
  - Independent development
  - Developer preferences
  - ...
  - All aspects result from autonomy
- Heterogeneity of metadata *and* data
- Two main approaches:
  - Try to resolve heterogeneity when needed
  - Enforce homogeneity/limit heterogeneity by establishing standards (not in this lecture)
    - No real solution to the problem
    - Only creates "spheres of homogeneity", any participants that have existing systems or requirements not conforming to the standards have to resolve heterogeneity locally

Technical Heterogeneity

- Refers to differences in the options to access data, e.g.
  - Communication protocols (HTTP, SOAP, …)
  - Exchange formats (binary, text, XML, …)
  - APIs (JDBC, ODBC, proprietary)
  - Query mechanism
    - Forms, canned queries
    - Query languages
  - Query language
    - SQL, XQuery, …

Data Model Heterogeneity

- Caused by the use of different data models among data sources
  - hierarchical, relational, XML, …
- Data models can have different expressiveness, e.g. support of
  - Inheritance
  - Types and degree of associations between entities/application concepts
  - Multi-valued attributes
  - Different atomic data types
- Mapping from semantically richer to poorer models in general results in a loss of information

Syntactic Heterogeneity

- Differences in the representation of identical facts
  - Binary representations (little/big endian, number formats)
  - Encodings (ASCII, ISO-8859-1, EBCDIC, Unicode, …)
  - Separators (Tab-delimited vs. CSV)
  - Textual representation
- Not to be mixed up with semantic heterogeneity!
- Usually easy to resolve (if used consistently)
- Examples:
  - "20070201" vs. "Februar 1st, 2007" vs. "02-01-07" vs. "1.2.2007"
  - "123.45" vs. "1.2345x10^2"

➡ *Data Fusion*

Structural Heterogeneity

- Caused by modeling identical application concepts differently using the *same* elements in the same data model
- Example - denormalized relational schema

Employee

EmpNo	Name	DoB	DeptNo
4711	Bob	1978-03-20	11
0815	Jane	1975-11-05	7
1234	Joe	1954-05-26	11

Department

DeptNo	Name
7	Sales
11	Accounting

⇔

EmpDept

EmpNo	Name	DoB	Deptname	DeptNo
4711	Bob	1978-03-20	Accounting	11
0815	Jane	1975-11-05	Sales	7
1234	Joe	1954-05-26	Accounting	11

- Easily resolved using relational operators:

```
SELECT e.EmpNo, e.Name, e.DoB, d.name as deptname, d.deptno
FROM Employee e, Department d WHERE e.deptno = d.deptno
```

Structural Heterogeneity (cont.)

- Example: inverted hierarchy

```
<bib>
  <book title=“a”>
     <author name=“x”/>
     <author name=“y”/>
  </book>
  <book title=“b”>
     <author name=“x”/>
  </book>
</bib>
```

⇔

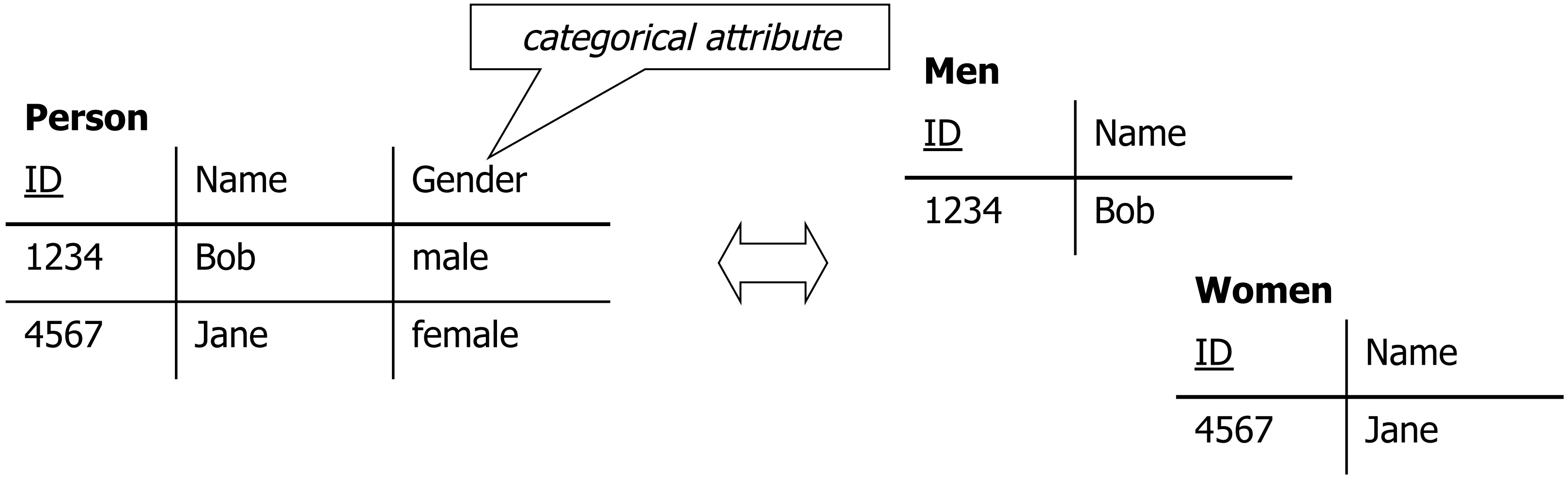
```
<bib>
  <author name=“y”>
    <book title=“a”/>
  </author>
  <author name=“x”>
    <book title=“a”/>
    <book title=“b”/>
  </author>
</bib>
```

- Easily resolved using XQuery

```
<bib> {
   for $a in distinct-values(doc("BookAuthor.xml")//author/@name)
   return <author name="{$a}"> {
      for $b in doc("BookAuthor.xml")//book
      where $b/author/@name = $a
      return <book title="{$b/@title}"/>
      } </author>
} </bib>
```

Schematic Heterogeneity

- Often considered a special case of structural heterogeneity
- Caused by modeling identical application concepts using *different* data model concepts of the same data model
- Example: *attribute value – relation name* conflict



categorical attribute

Person

<u>ID</u>	Name	Gender
1234	Bob	male
4567	Jane	female

Men

<u>ID</u>	Name
1234	Bob

Women

<u>ID</u>	Name
4567	Jane

- Problems of this kind cannot be resolved *generically* with SQL
  - How to handle an unknown/variable number of values for categorical attributes?

Semantic Heterogeneity

- "Semantics" = interpretation of data and metadata
- Different representation of identical application concepts, (e.g. synonyms)
- Identical representation of different application concepts (e.g. homonyms)
  - e.g. Lotus (the car) vs. Lotus (the flower)
- Ambiguities – unclear whether two elements refer to the same concept (are synonyms) or refer to broader/narrower terms (hypernyms)
  - hypernym or synonym?
    - car – (motor) vehicle
    - person – employee
    - product – item
  - decision depending on context
- Perhaps *the* biggest challenge in II
- Resolving semantic heterogeneity is a prerequisite for many integration tasks
- Many attempts to automate
  - ➡ *Schema Matching*

Data Integration Middleware

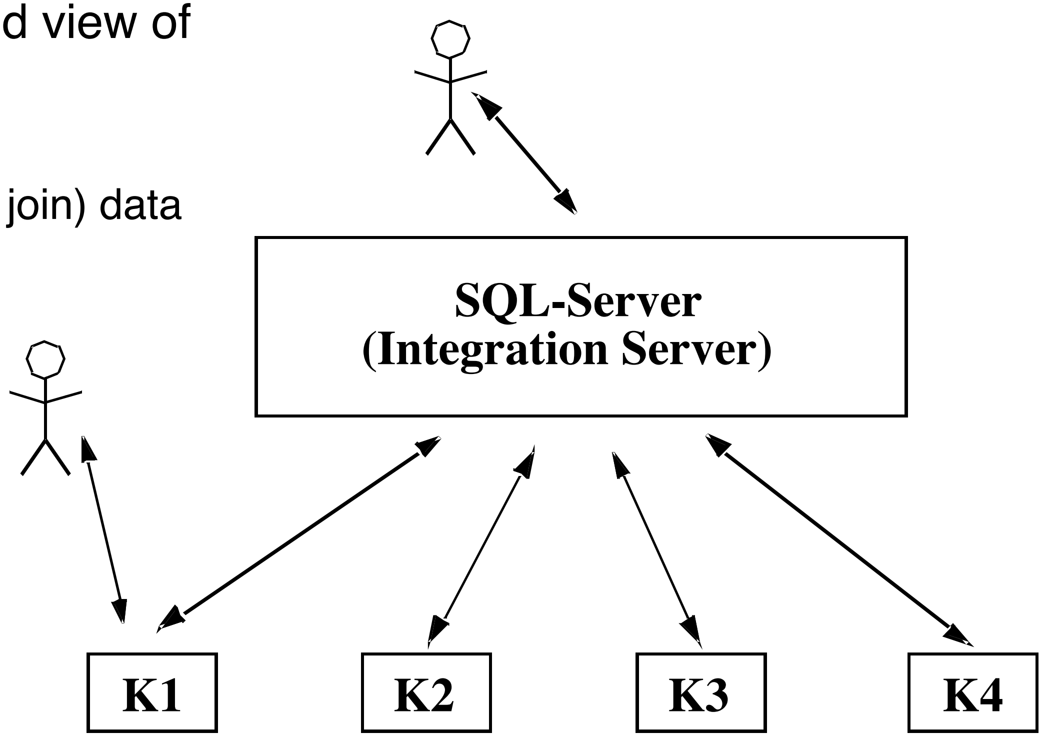
- Traditional Middleware
  - supports access to multiple data sources within the same application, transaction
    - directly (using DB-gateways)
    - indirectly (by invoking distributed application components)
  - but fails to provide data integration
    - no means to analyze/query data from multiple sources within the same statement

      ```
      SELECT *
      FROM Source1-table T1, Source2-table T2
      WHERE T1.a1 = …
                AND
                T1.a2 = T2.a1
      ```

    - does not help to overcome data heterogeneity
- Two architectural approaches to achieve data integration
  - materialized integration: replication, data warehousing
  - virtual integration: federated DBMS, multi-database systems

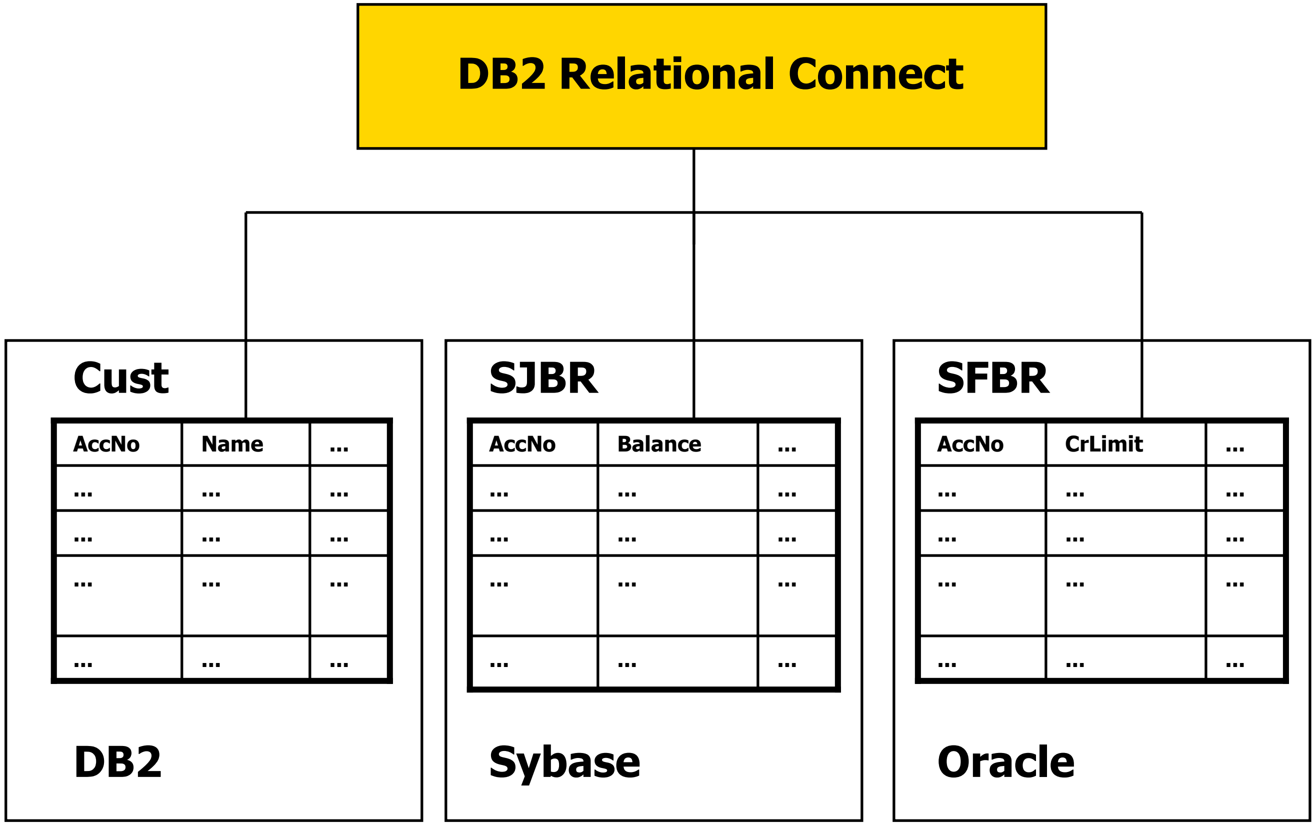
Data Federation: Virtual Integration

- Goal: homogeneous, integrated view of data from multiple sources
  - a single (logical) database
  - a single query may collect (or join) data from multiple sources
- Data Federation requires
  - Wrapper/mediator technology
  - Data and schema integration mechanisms



Example - DB2 Relational Connect

```
Select  *
From    Cust, SJBR, SFBR
Where   Cust.Acct No = SJBR.Acct No
And     SJBR.Acct No = SFBR.Acct No
```



DB2 Relational Connect

Cust

AccNo	Name	...
...	...	...
...	...	...
...	...	...
...	...	...

DB2

SJBR

AccNo	Balance	...
...	...	...
...	...	...
...	...	...
...	...	...

Sybase

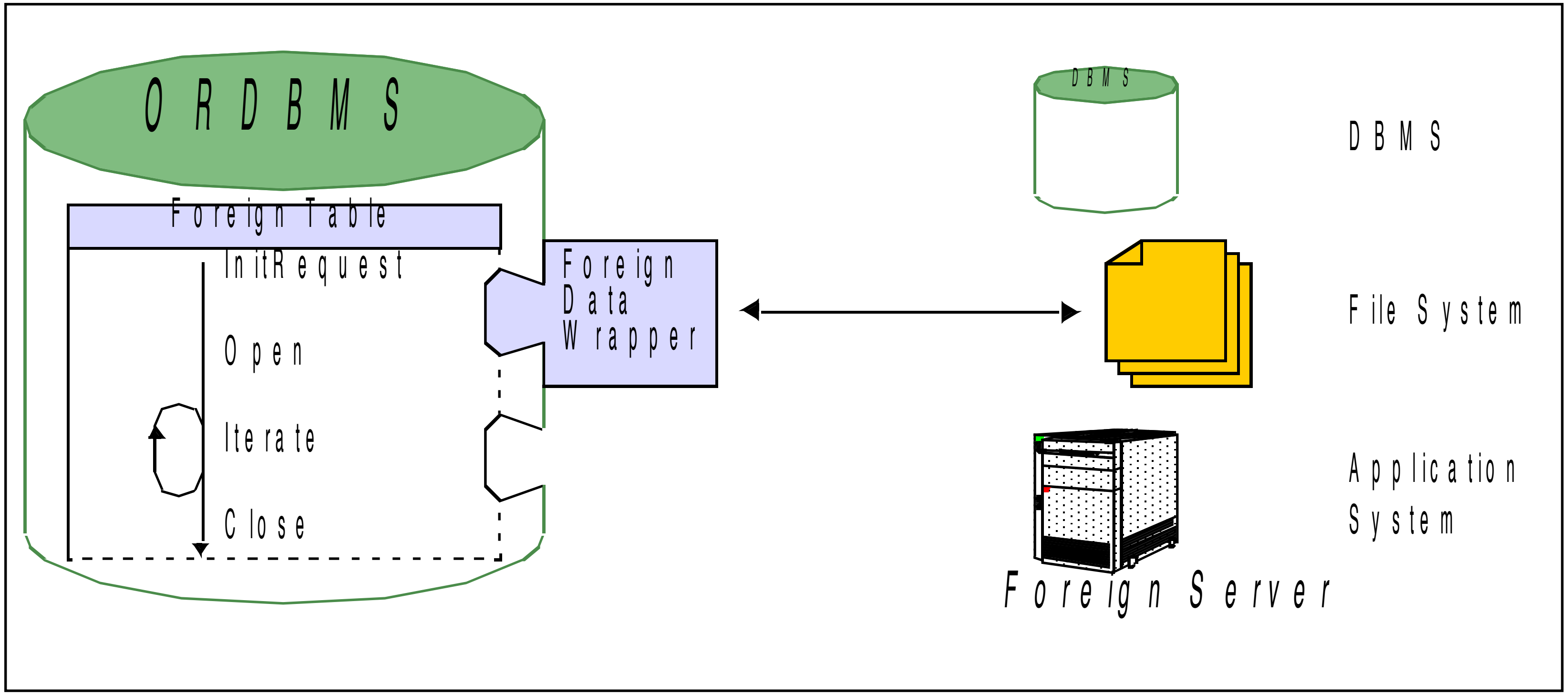
SFBR

AccNo	CrLimit	...
...	...	...
...	...	...
...	...	...
...	...	...

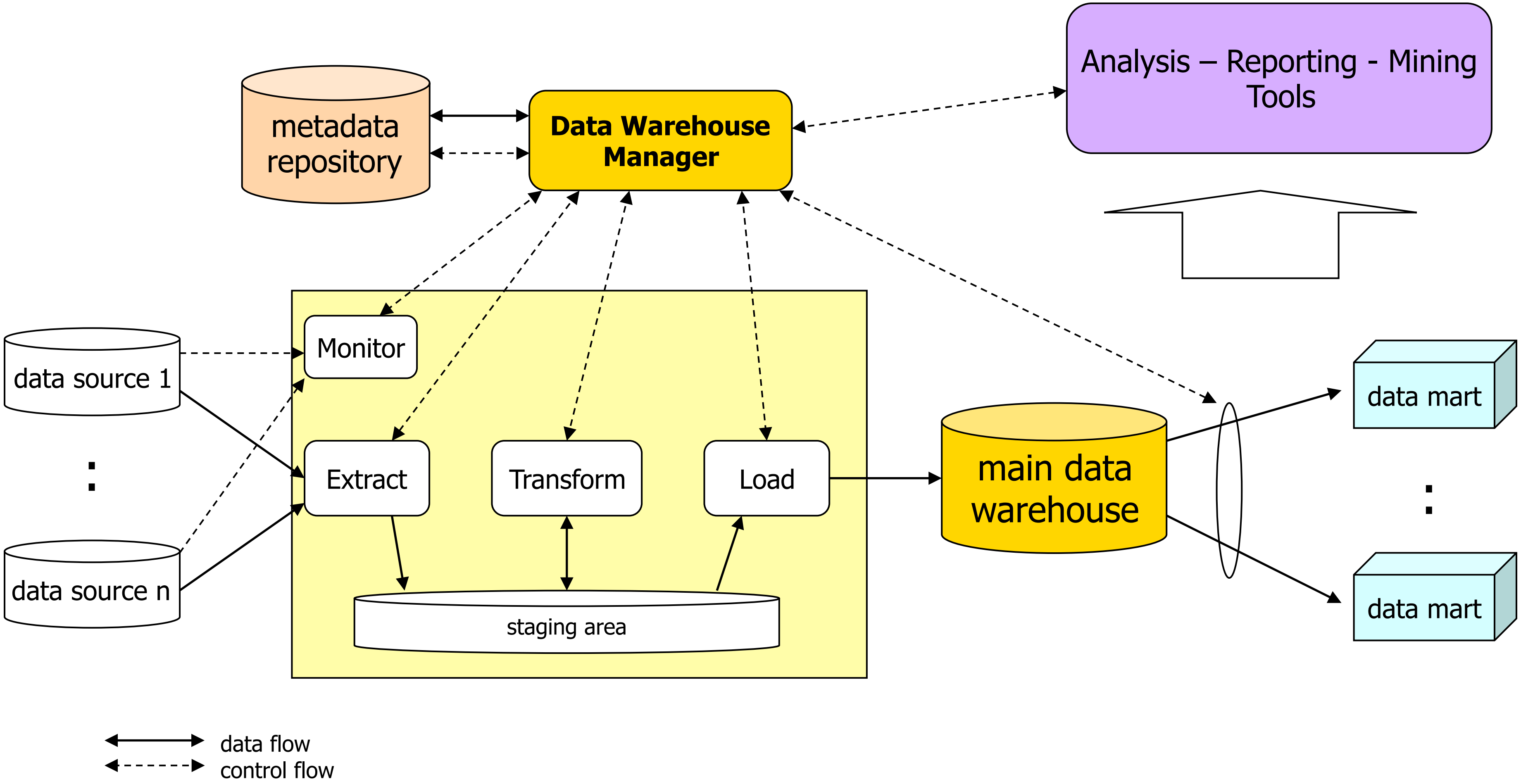
Oracle

Standard – SQL/MED

- ‘Foreign Data Wrapper’ in ‘SQL/MED’



Data Warehousing Architecture



© Prof.Dr.-Ing. Stefan Deßloch

Integration Process

- Schema Matching
  - Find inter-schema correspondences
- Schema Mapping
  - Based on correspondences
  - Define how to "translate" one schema into another
    - implies data transformation
- Schema Integration
  - Based on correspondences (and mapping)
  - Define an integrated, global/federated schema

➔ **Integration Plan!**

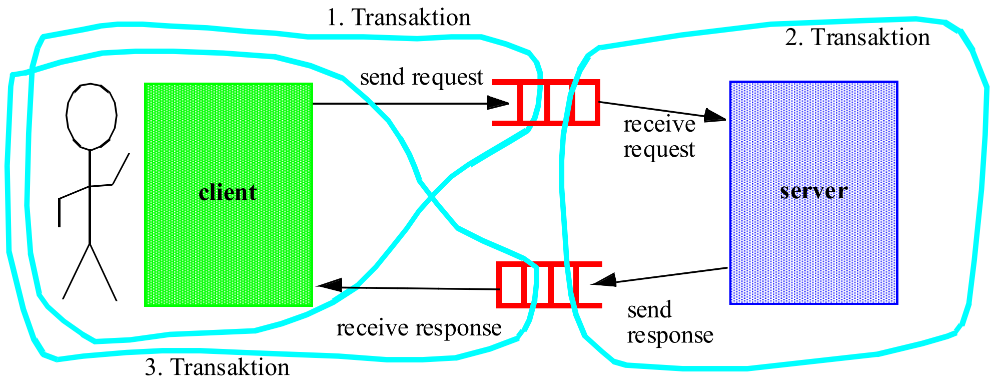
- Integration plan can then be “implemented” using middleware for virtual or materialized data integration

Enterprise Application Integration

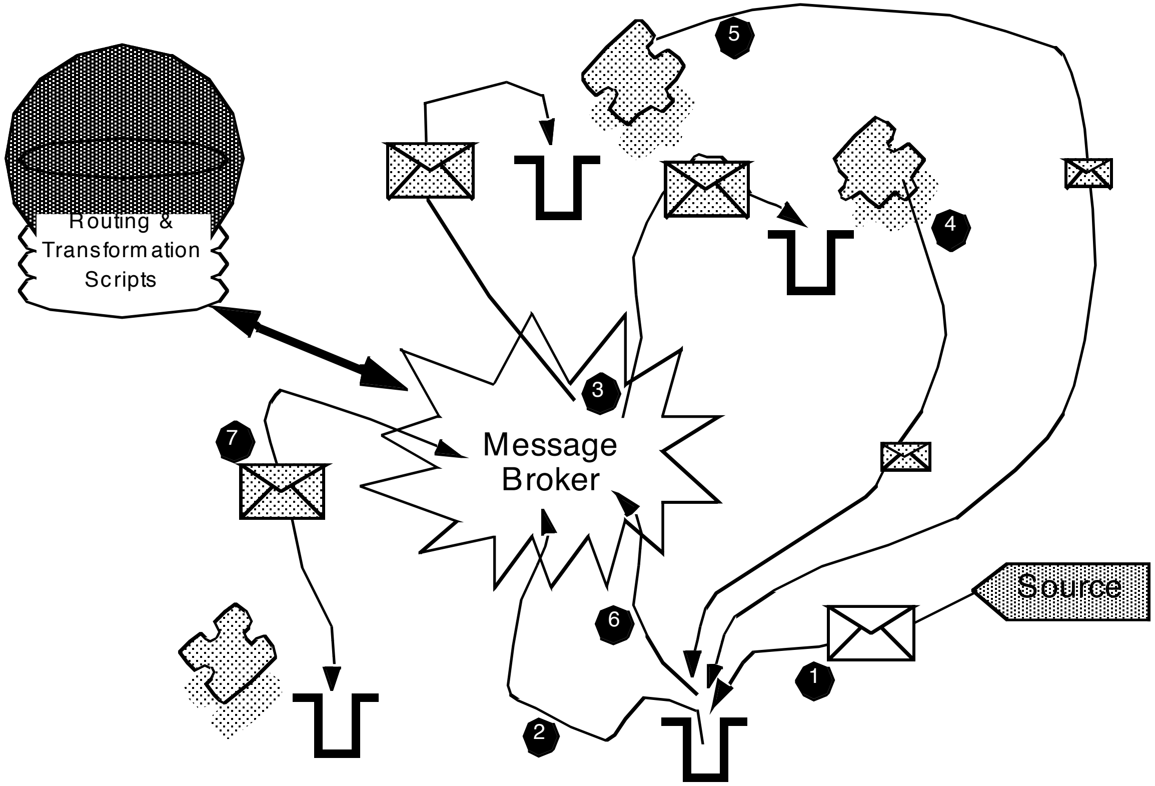
- Focus on application integration within an enterprise (vs. development of new application)
  - integration across different middleware platforms
  - major shift towards asynchronous interactions (**M**essage-**O**riented **M**iddleware)
- Message Brokers
  - based on MOM
  - hub-and-spoke (instead of point-to-point)
  - publish and subscribe model to link applications together
- Business Process Modeling and Workflow Management Systems
  - make integration logic explicit, easy to modify/extend
  - "programming in the large"

Message-Oriented Middleware (MOM)

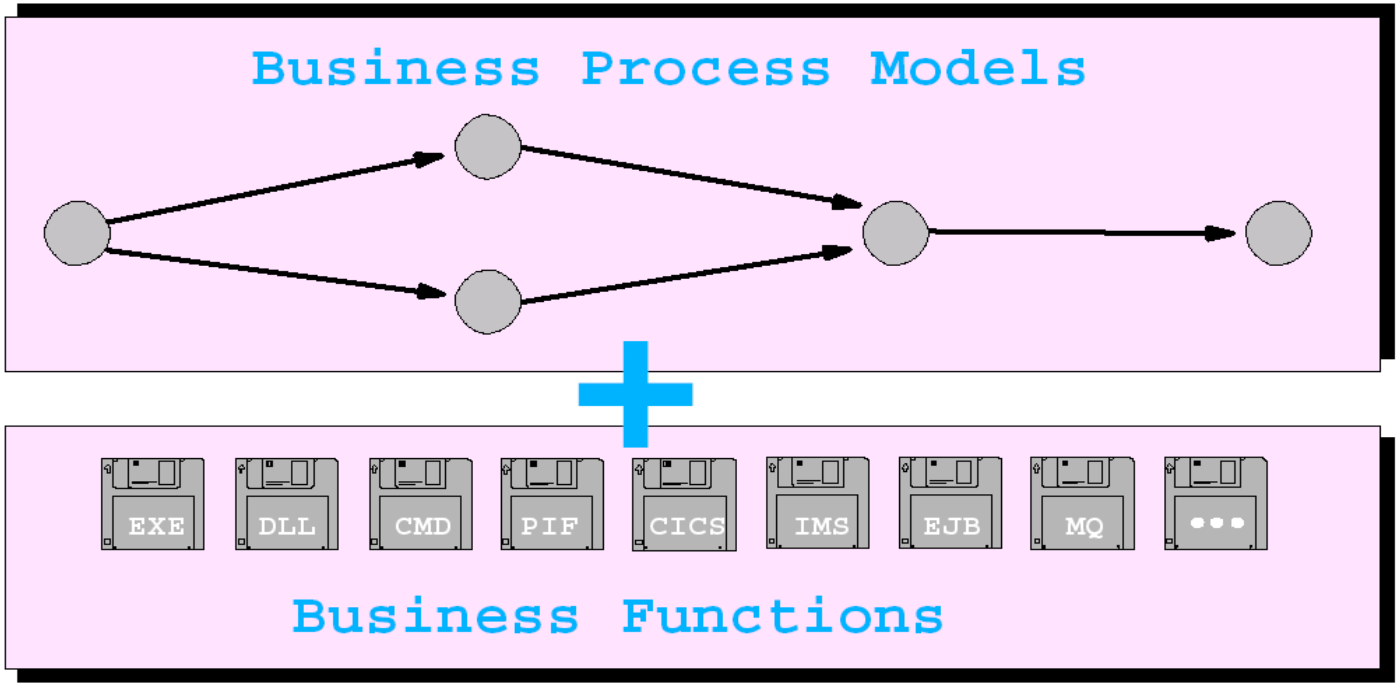
- Message-oriented interoperability
  - programming model: asynchronous message exchange
- Support for persistent, transactional message queues
  - asynchronous transactions
  - reliable messaging
- Optimizing throughput, not response time
- Loosely-coupled application components
  - "client" not blocked during request processing
  - "server" may chose request processing time more flexibly
    - may not even be available at request enqueue time



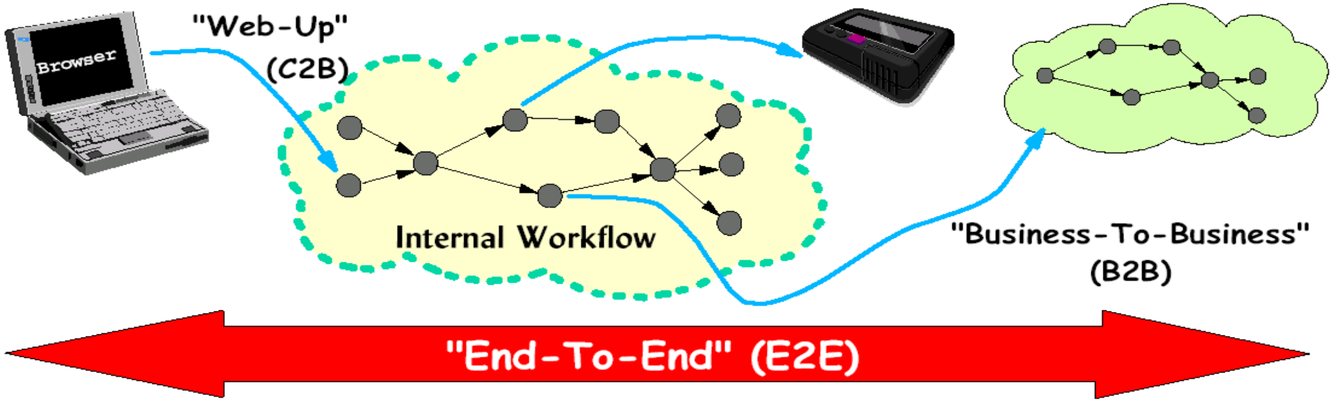
Message Brokering – Processing Model



Workflow-Based Applications: Structure

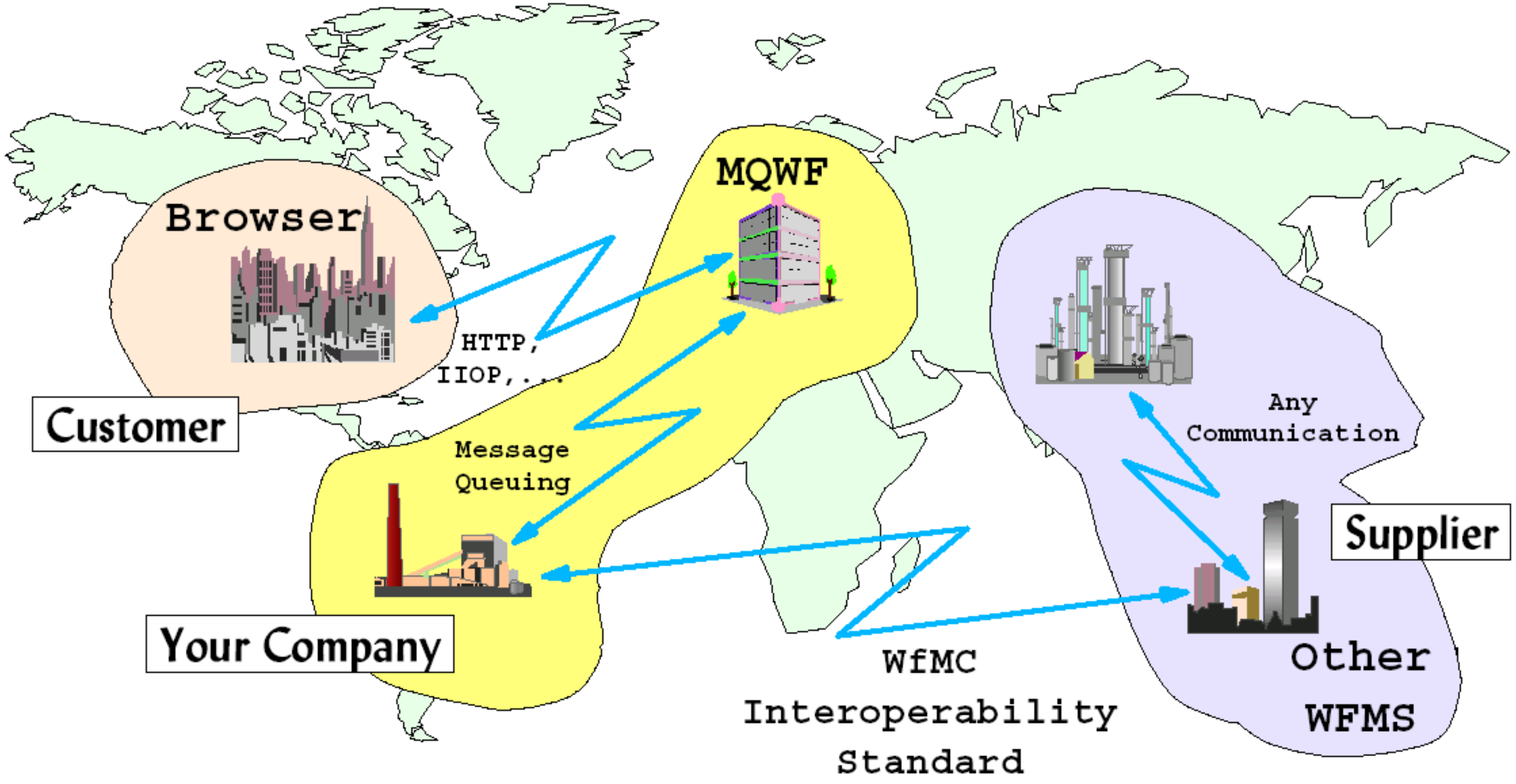


Workflows And External Communications



- Customers invoke company's applications to perform certain steps of the business process
  - E.g. place on order, inquire status,...
  - Company's applications must get a browser-based front-end for that purpose ("web-up")
- Workflow activities may directly communicate with the outside
  - Send e-mail, faxes, messages,...
- Workflow activities may trigger actions in another company
  - Simple invokation of program or start of another workflow ("subprocess" from invokers point-of-view)
  - Such "business-to-business" scenarios are the base for realizing sophisticated "supply chains"

Virtual Enterprise: Scenario

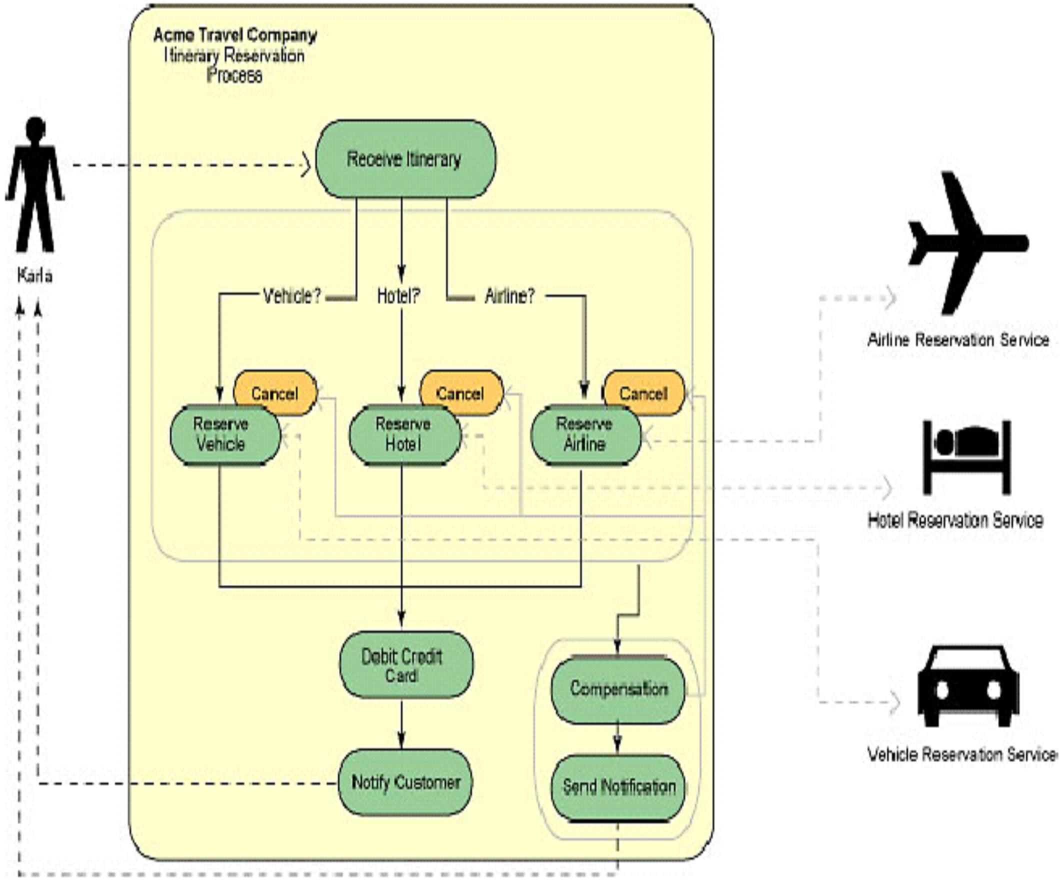


Business-To-Businesss (B2B) Integration

- Goal: facilitate interaction among trading partners, across companies
  - Establish relation between processes of different enterprises
  - Predominant are relation to suppliers, and customer relations to other enterprises like industrial consumers, retailers, banks
- Traditional B2B has focused on well-defined, standard message formats and protocols (e.g., RosettaNet, cXML)
  - Ad hoc B2B occurs today via XML over HTTP
- How to publish business functions to customers, partners and suppliers?
  - E.g. access to reservation systems, quote systems
  - Programmatic access to a service, independent of underlying implementation and client software
- Web services, service-oriented architectures play a dominant role!

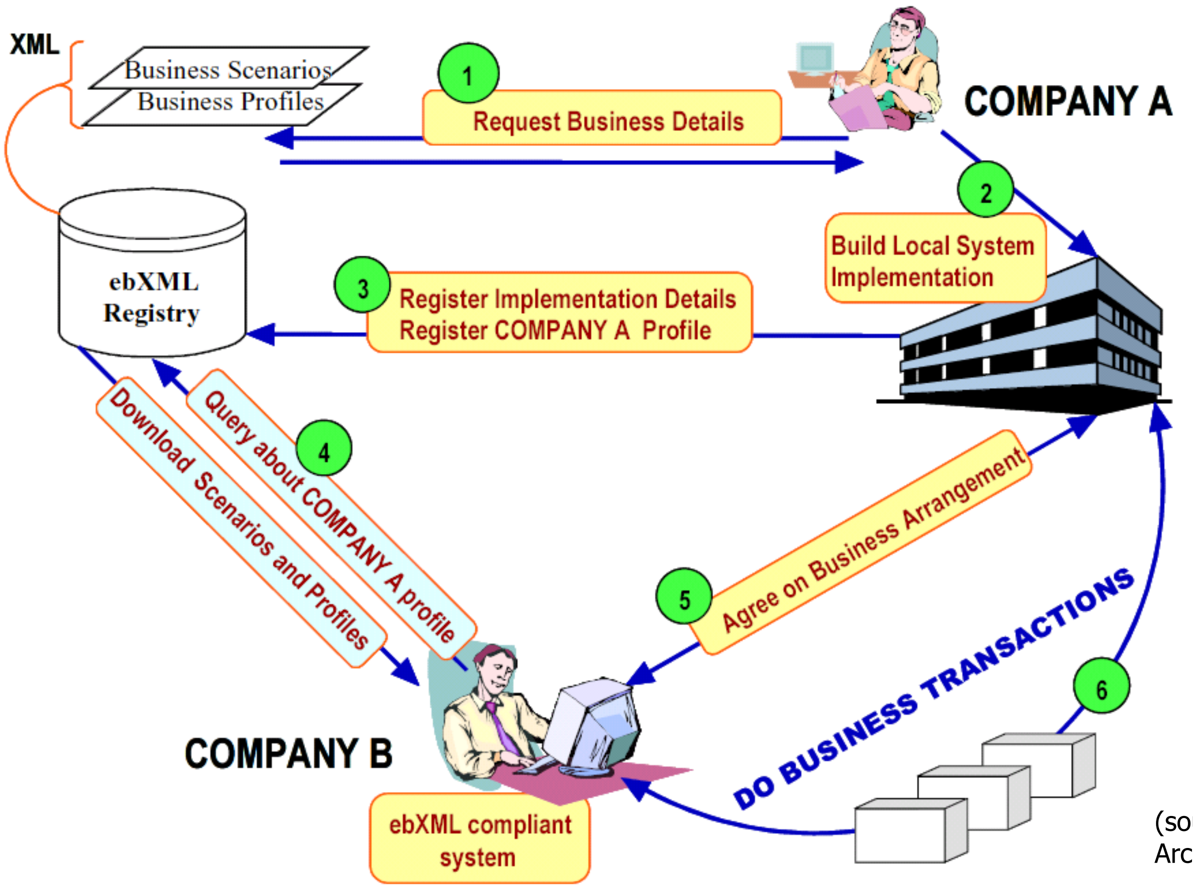
Web Services & Business Processes

- Business process making use of web services
- Business process externalized as a web service
- Long-running transactions
- Compensation
- Correlation
- Dynamic Binding of business partners and web services



e-Business Collaboration

- Example: ebXML


XML
Business Scenarios
Business Profiles
1
Request Business Details
COMPANY A
ebXML Registry
2
Build Local System Implementation
3
Register Implementation Details
Register COMPANY A Profile
Download Scenarios and Profiles
Query about COMPANY A profile
4
5
Agree on Business Arrangement
DO BUSINESS TRANSACTIONS
6
COMPANY B
ebXML compliant system

(source: ebXML Technical Architecture Specification)

Summary

- Middleware
  - supports the development, deployment, and execution of complex information systems
  - facilitates interaction between and integration of applications

  across multiple distributed, heterogeneous platforms and data sources
- Major challenges: distribution, autonomy, heterogeneity
  - different forms of (data) heterogeneity
- Data/Information Integration
  - integrated access to (heterogeneous) data originating from multiple sources
- Enterprise Application Integration
  - integration of (heterogeneous, coarse-grained) applications within an enterprise (vs. development of new application)
  - integration across different middleware platforms
- Business-to-business Integration
  - support interactions, integration of business processes among trading partners, across company boundaries
  - foundation for e-business, e-commerce