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 12 – Information Systems Integration

IS Integration - Approaches

- Data/Information Integration
  - integrated access to (heterogeneous) data originating from multiple sources
    - queries range over data 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, typically based on MOM
- Business-to-business Integration
  - support interactions, integration of business processes among trading partners, across company boundaries
  - foundation for e-business, e-commerce
  - typically based on middleware for web service orchestration/choreography

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, object-relational, object-oriented, 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. "1.2345×10^2"
- ➡ *Data Fusion*

Structural Heterogeneity

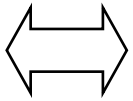
- Caused by modeling identical application concepts differently using the *same* modelling concepts in the same data model
- Example - denormalized relational schema

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

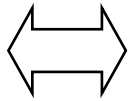
- Easily resolved using relational operators:

```
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>
```



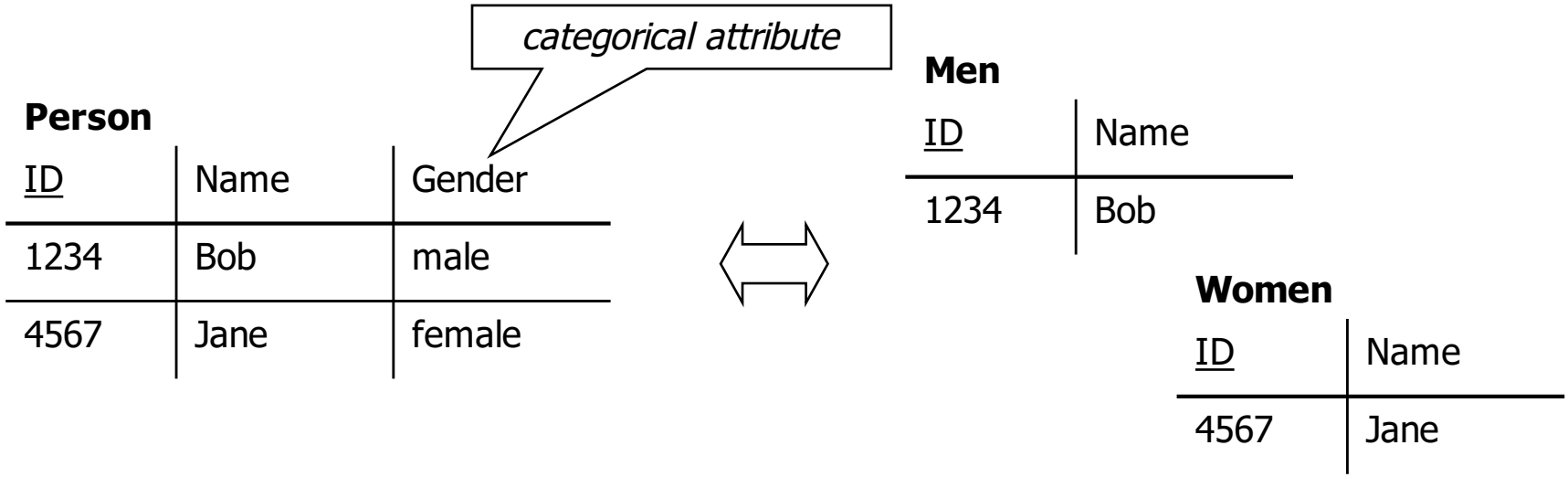
```
<bib>
  <author name="y">
    <book title="a"/>
  </author>
  <author name="x">
    <book title="a"/>
    <book title="b"/>
  </author>
</bib>
```

- 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



- 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

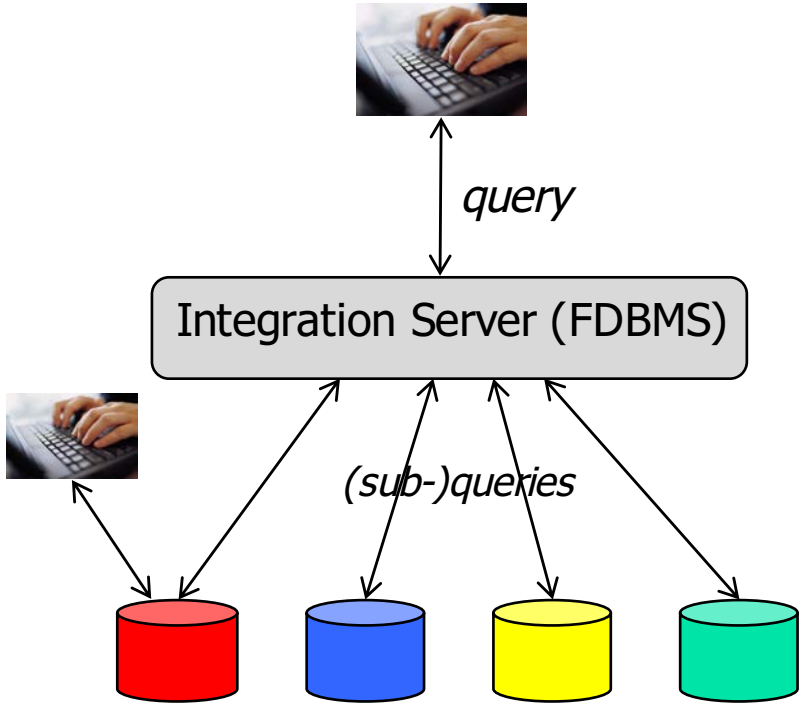
- Traditional Middleware (shortcomings)
  - supports access to multiple data sources within the same application, transaction
    - directly (using DB-gateways)
    - indirectly (by invoking distributed application components)
  - but fails to provide data integration
    - no means to analyze/query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

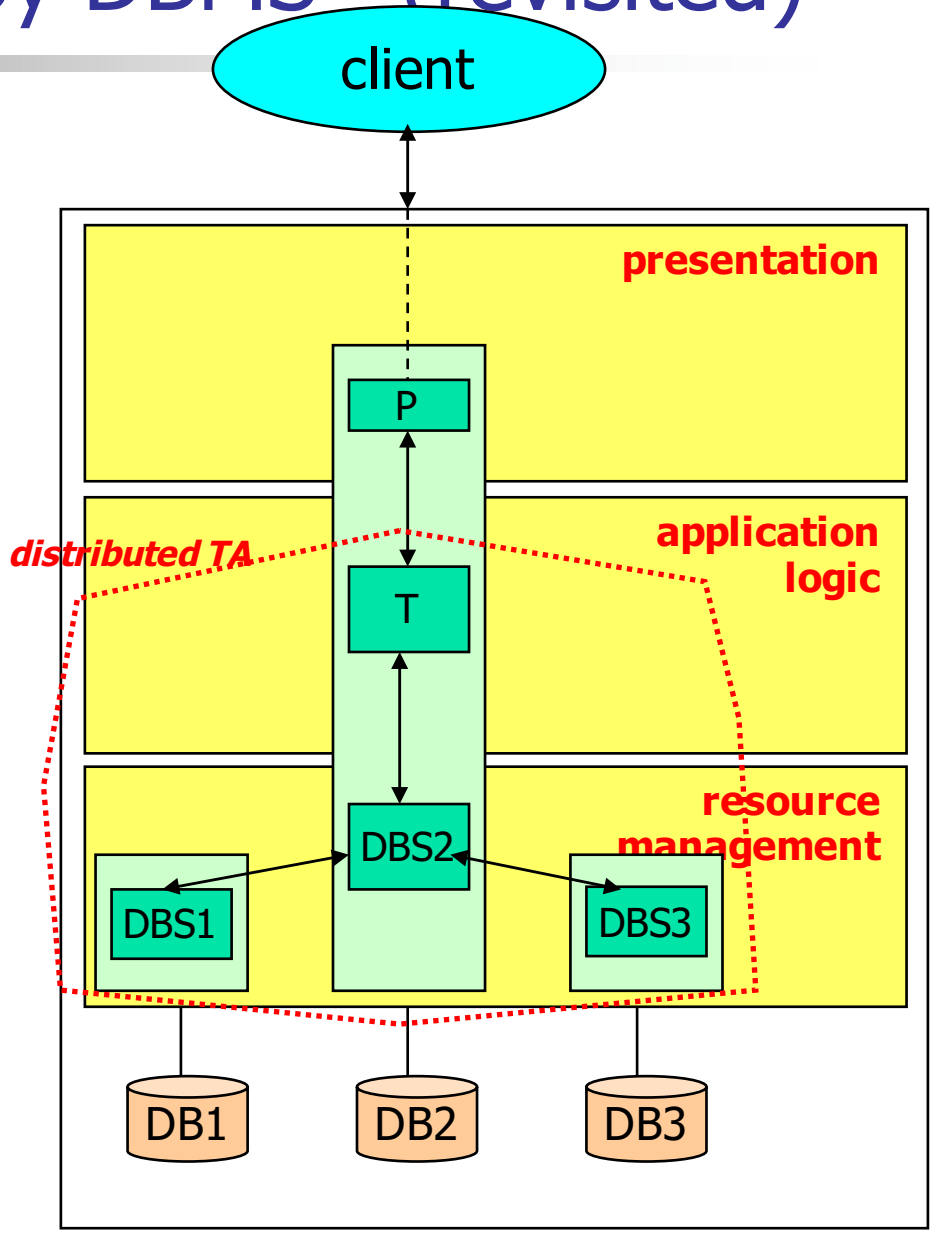
Data Federation: Logical (Virtual) Integration

- Goal: homogeneous, integrated view of data from multiple sources
  - illusion of a single (logical) database
  - a single query may collect (or join) data from multiple sources
- "On-Demand" Data Integration
  - data stays where it is (in the sources)
    - not copied into a new DB
  - data is transformed/integrated at query time
    - integration server combines results from data source queries
- Data Federation requires
  - Wrapper/mediator technology
  - Data and schema integration mechanisms

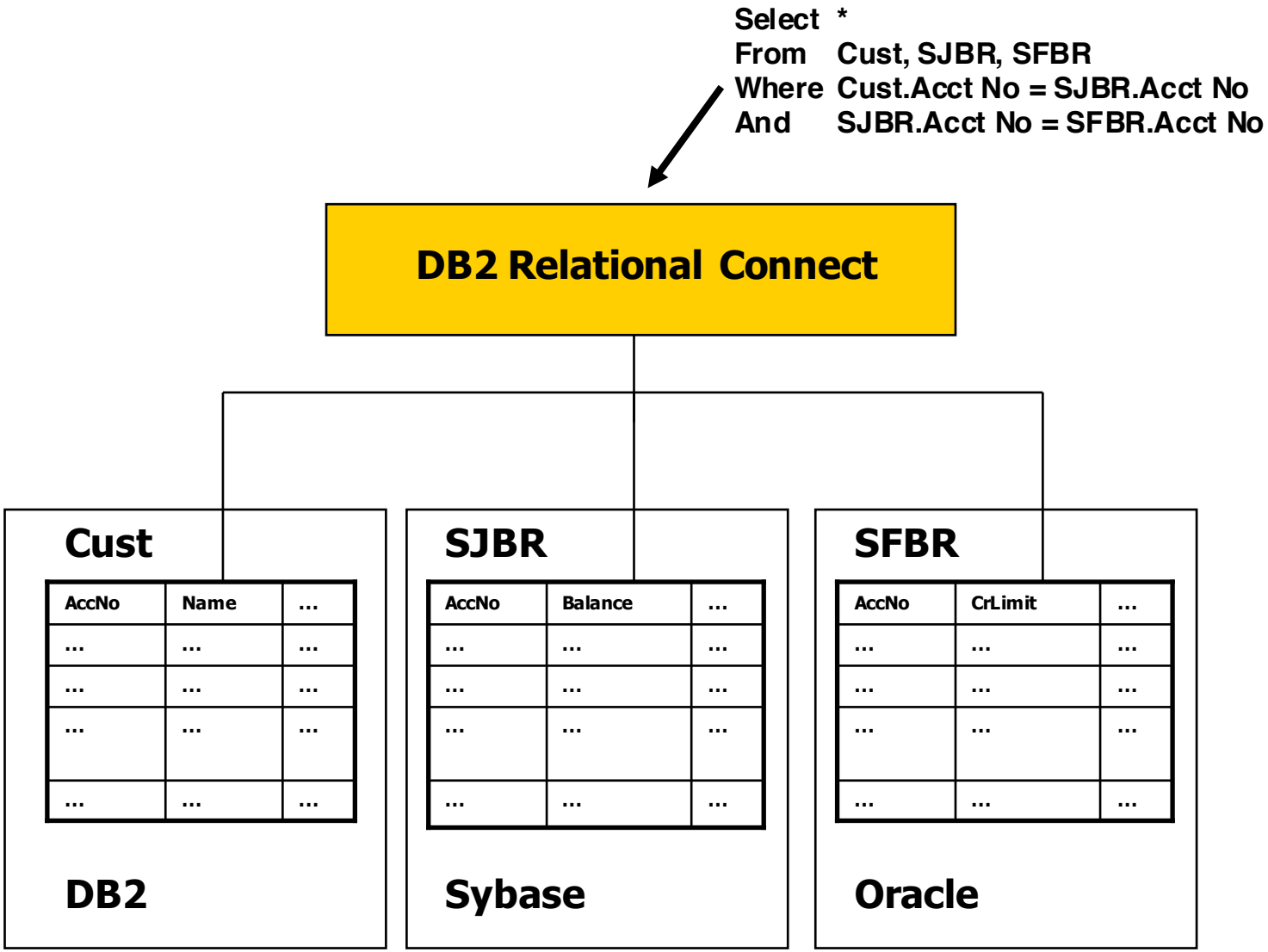


Distribution Controlled by DBMS (revisited)

- 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

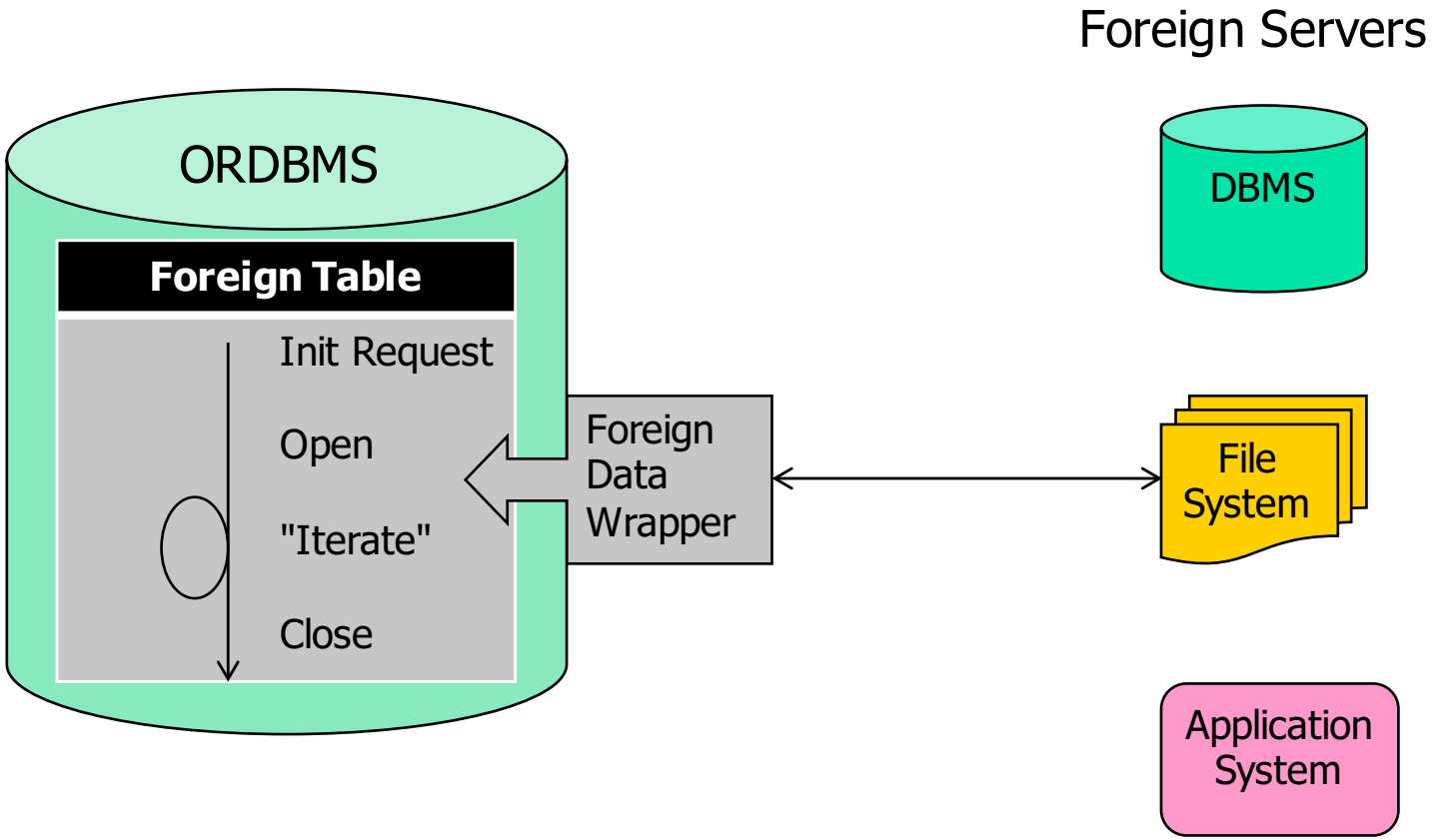


Example - DB2 Relational Connect

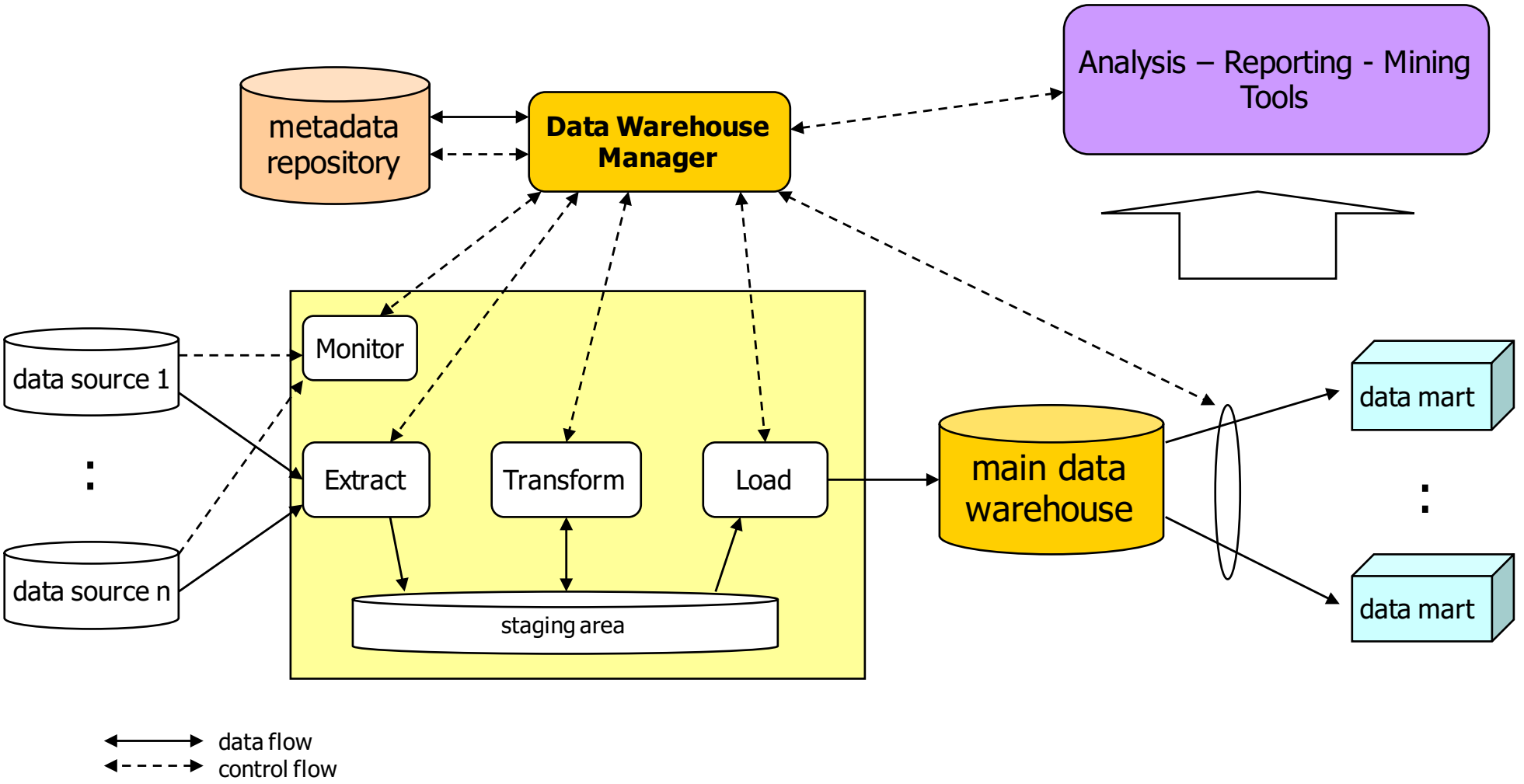


Standard – SQL/MED

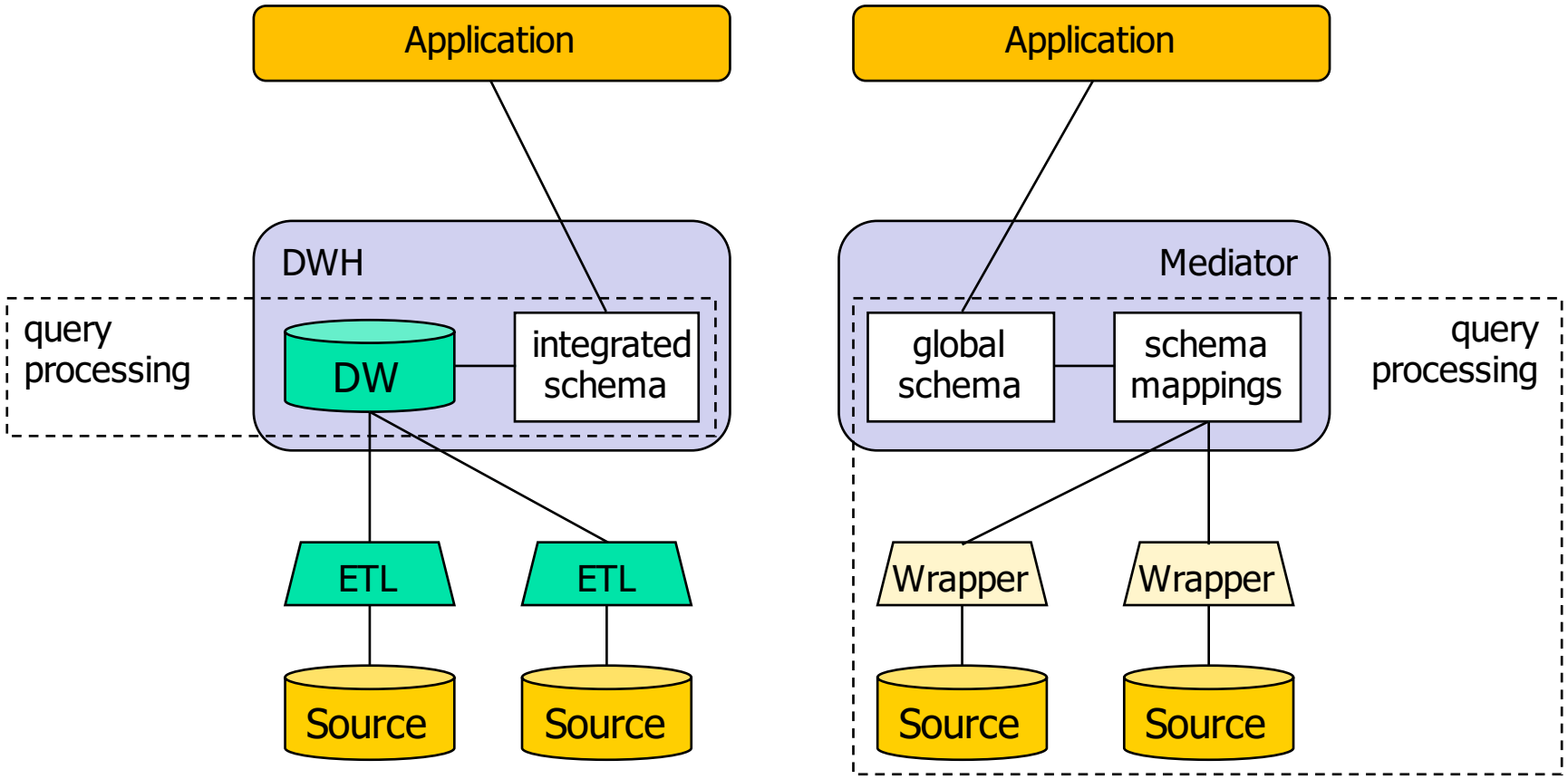
- 'Foreign Data Wrapper' in 'SQL/MED'



Data Warehouse: Physical (Materialized) Integration



Materialized and Virtual Integration



Pros and Cons of Materialized vs. Virtual DI

	Materialized	Virtual
data freshness	low: update frequency	high: always fresh
query response time	low: local processing	high: distributed proc.
complexity	low: like DBMS	high: distr. query planning
loss of autonomy	providing load data	query processing
query support	unlimited, full SQL	may be limited (by source)
read/write	both (on DWH)	typically read-only
storage resources	high: full data	low: only meta data
load on sources	load/update: planned	query: not planned
data cleaning	possible	not possible

Bridging/Resolving Heterogeneity

- Real-world integration scenarios suffer from all kinds of heterogeneity
- Middlewre and the primary issues they address:
  - Wrappers (data model heterogeneity, technical heterogeneity, syntactic heterogeneity)
  - Mediators/federated DBMS (technical heterogeneity, structural heterogeneity, distribution)
  - Multi-database languages (schematic heterogeneity, technical heterogeneity, distribution)
  - DB Gateways (technical heterogeneity)
  - ETL tools (structural heterogeneity, technical heterogeneity, syntactic heterogeneity)
  - ⇨ focus on data access/transformation infrastructure (i.e., as a runtime platform)
- Further general integration techniques needed
  - Schema Matching and Integration (semantic heterogeneity, structural heterogeneity)
  - Data Cleaning/Fusion (syntactic heterogeneity, semantic heterogeneity (in data))
  - ⇨ focus on integration planning

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

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
  - goal: integrated access to (heterogeneous) data originating from multiple sources
  - materialized integration extracts data from sources and stores it in an integrated database for query processing, data analysis
  - virtual integration leaves data in the sources and performs complex, distributed query processing for data access
  - both approaches have pros and cons
- Information integration process
  - general techniques, independent of middleware plattforms