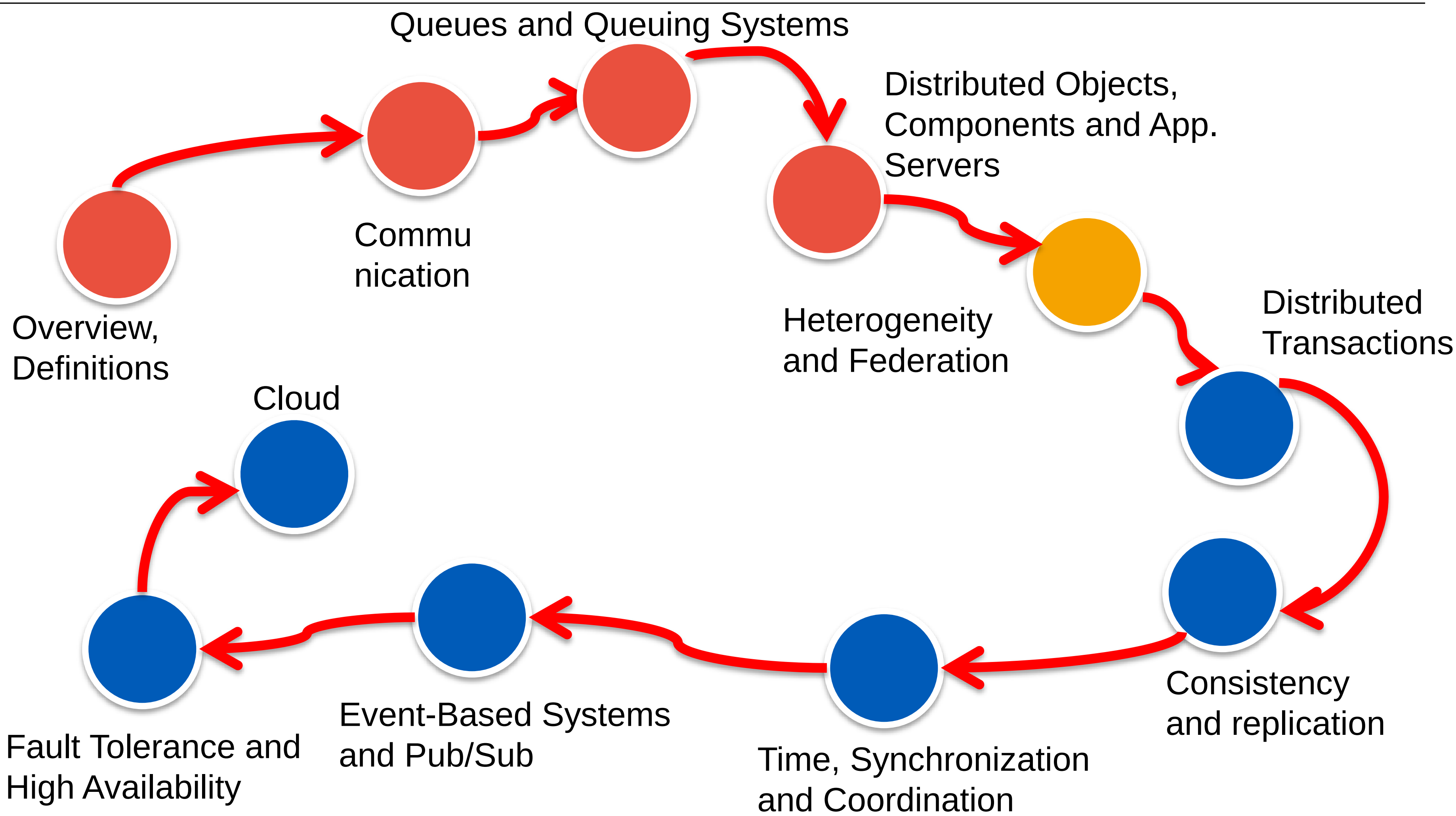
Middleware
5. System Federation

I. Petrov, A. Buchmann
Wintersemester 2011/2012

Topics



Queues and Queuing Systems
Distributed Objects, Components and App. Servers
Overview, Definitions
Commu nication
Heterogeneity and Federation
Distributed Transactions
Cloud
Consistency and replication
Fault Tolerance and High Availability
Event-Based Systems and Pub/Sub
Time, Synchronization and Coordination

Topics

- System federation
- Heterogeneity
- Federated Databases
- Information integration
- Wrappers/Mediators

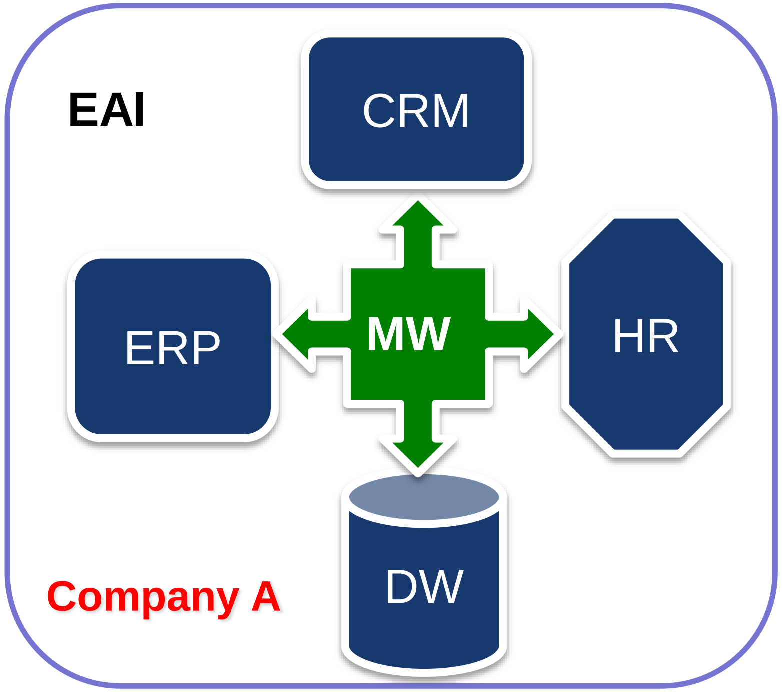
Reading for THIS Lecture

- The slides for the lecture are based on material from:
  - M. Tamer Özsu and Patrick Valduriez **Principles of Distributed Database Systems** (3nd Ed.). Prentice-Hall. 2011
    - Section 1.7.10, Chapter 4, 9
  - Gio Wiederhold. 1992.
    Mediators in the Architecture of Future Information Systems. Computer 25(3),pp.38-49
  - L. M. Haas, E. T. Lin, and M. A. Roth. 2002.
    Data integration through database federation. IBM Syst. J. 41, 4 (October 2002)
  - David Linthicum
    Enterprise Application Integration, Addison-Wesley 2007.
    - Chapter 2, 3, 4; pp 21-25
  - Luke Hohmann
    Beyond Software Architecture: Creating and Sustaining Winning Solutions**.** Addison Wesley, 2003
    - Chapter 8
  - Martin Fowler. Patterns of Enterprise Application Architecture. Addison-Wesley . 2002
    - Chapter 10, 14, 18

System Federation and Integration

Created with wordle.net based on:
P. Bernstein. Middleware. CACM, Feb. 1996

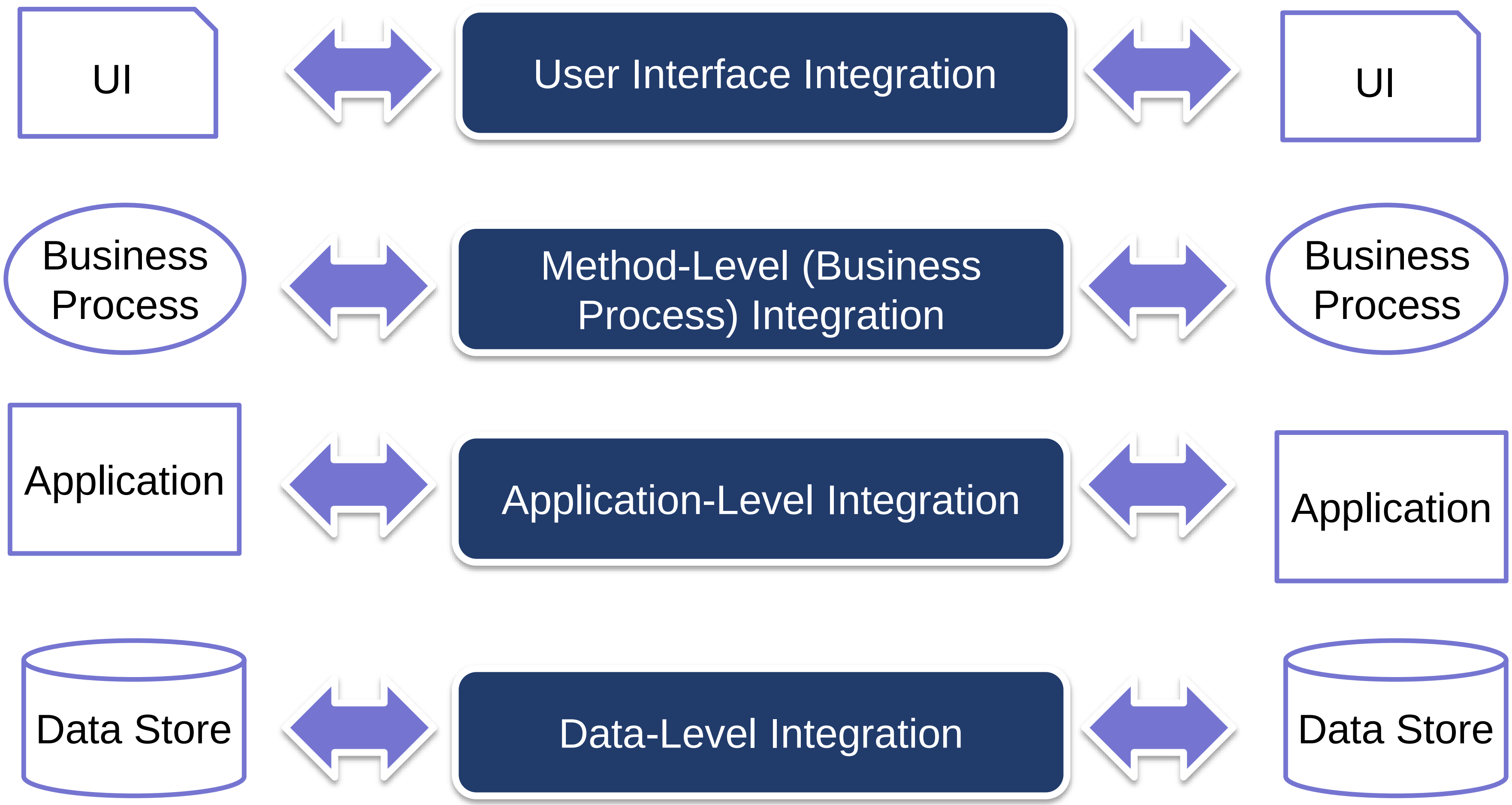
Enterprise Application Integration (EAI)



- EAI deals with the integration of applications
- Always within the boundaries of one enterprise
- Systems are integrated through middleware

- Problem Space:
  - **Autonomy**
  - **Heterogeneity**
  - **Distribution**

Types of Integration



Application Interface Integration

TECHNISCHE UNIVERSITÄT DARMSTADT

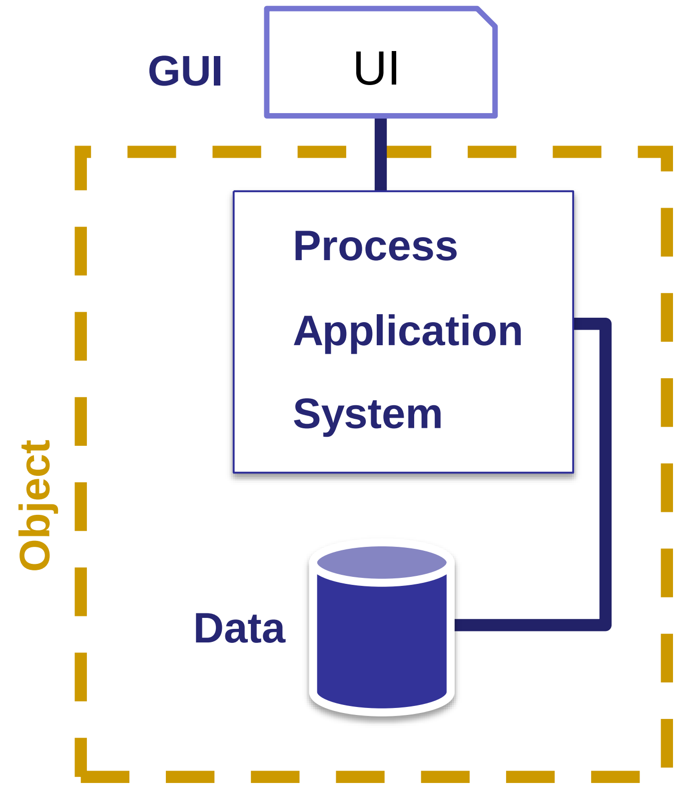
- Leverages interfaces exposed by custom or packaged applications
  - through exposed interfaces business information is accessed and applications can share business logic
- Most applicable to packaged systems like SAP or PeopleSoft/Oracle
- Must extract data and process information
- Message brokers are best solution

Reading: David Linthicum, Enterprise Application Integration, Addison-Wesley 2007. Chapter 3

DVS

Method-oriented Integration

- Sharing of business logic that exists in the enterprise
  - e.g. updating customer record with one method from multiple applications
- 2 basic approaches:
  - create a shared set of application servers (TPM, application servers)
  - share methods already existing inside applications using method-sharing technology (distributed objects)



Reading: David Linthicum, Enterprise Application Integration, Addison-Wesley 2007. Chapter 4

Process-oriented Integration

- Places an abstract business layer on top of existing systems
- Abstraction of business processes to one common understanding
- Leverages other basic integration approaches (data, API, method)-oriented
- Tendency when moving to service-based architectures (e.g. Web-services)

Data-Oriented Integration

TECHNISCHE UNIVERSITÄT DARMSTADT

- Integration occurs through data extracted from one database (information system), processing of that data and insertion in another database
  - low cost since applications don‘t need to be changed and redeployed
  - most commonly used
  - complexity often appears lower than it is because of size (hundreds or thousands of tables) and semantic and schematic heterogeneity

Reading: David Linthicum, Enterprise Application Integration, Addison-Wesley 2007. Chapter 2

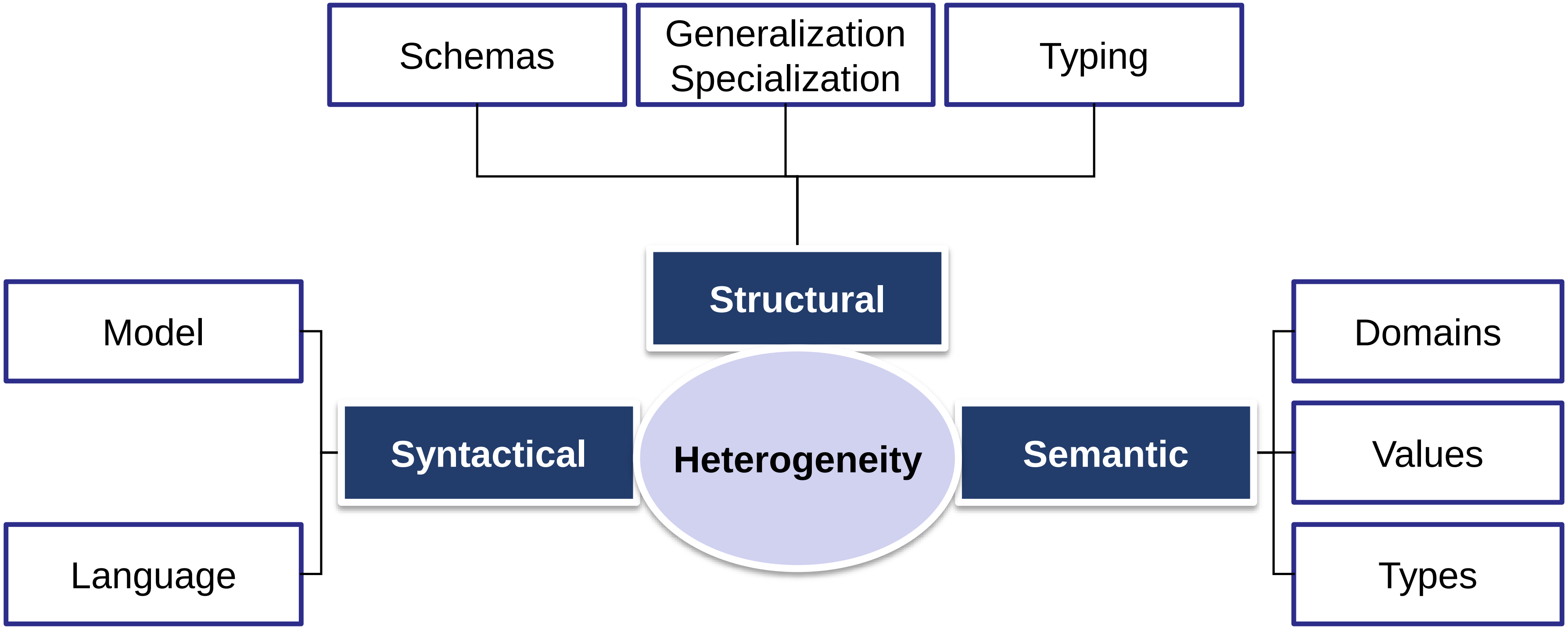
Portal-oriented Integration

- Use of one common interface (usually browser based) to access multiple applications
- Superficial integration at the user interface level
- Avoids more expensive back-end integration but has similar problems as data-oriented integration
- Burden of interpretation often placed on users
- Legacy

Heterogeneity

Created with wordle.net based on:
P. Bernstein. Middleware. CACM, Feb. 1996

Heterogeneity



Semantic Heterogeneity

- When the same information is represented in two separate Apps in structurally different but formally equivalent ways. Main reasons:
  - Independent design and evolution of autonomous Apps
  - Rich set of modeling constructs
  - Different modeling alternatives

- Spectrum of heterogeneity
  - Domain conflicts - the same entity is described differently in different domains
    - **Paul** is known as **p123** in domain A and **paul** in domain B
    - By using semantic equivalences and context dependence
    - Non-singular transformation (e.g. conversion rates)
  - Naming conflicts - Objects may be referred to in a different manner
    - The same attribute has different labels - Attribute **name** versus **lastname**
  - Type conflicts - Systems represent low level atomic values differently
    - Temperature: integer in SysA and float in SysB

Structural Heterogeneity

TECHNISCHE UNIVERSITÄT DARMSTADT

- Different structural organization used to represent the same concepts
  - AddressType represented as
    - Complex object
    - Single String attribute

- Overcoming Structural Conflicts
  - Involves decomposition or composition
    - Type level: Type1 ➔ Type2 & Type3
  - Structural transformation!
    - Complex, lossy
    - Data ←→ Metadata | Metadata ←→ Metadata | Data ←→ Data

Example: XML-Relational Mapping

- **Rel → XML**
- Data types
  - Trivial
- Integrity Constraints (e.g. primary keys)
  - Structure, requires XML Schema
- Operations
  - none in XML

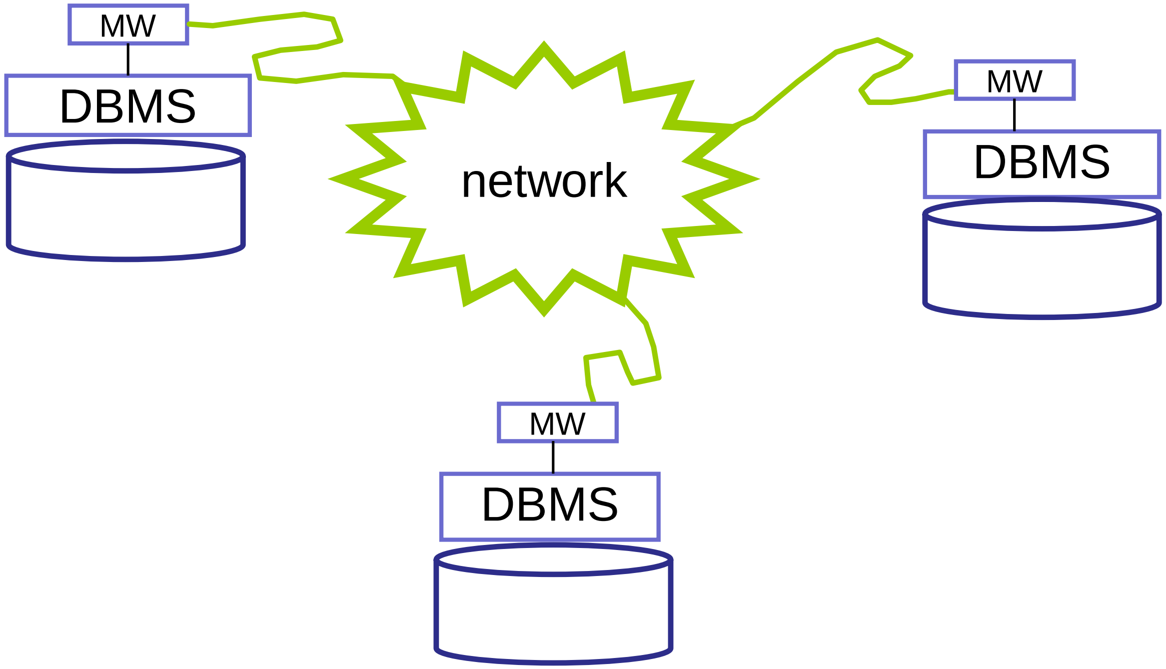
- **XML→Rel**
- Data Types
  - Lossy, no/non-equiv. types, collections…
- Integrity Constraints
  - none in XML (facets?)
- Operations
  - requires generally Xquery

Data Model Level	***Relational Concepts***	***XML Concepts***
	Relation → Attribute	Element Type → Attribute
Schema Level	***Relational Schema***	***DTD / XML Schema (optional)***
	Relation A → Attribute X Relation B → Attribute Y … …	Element Type a → Attribute x Element Type b → Attribute y … …
Instance Level	***Relational Database***	***XML Document***
	Tuple → Value	Element → Attribute Element Value Attribute Value

[A generic load/extract utility for data transfer between XML documents and relational databases, R. Bourret, C. Bornhovd, A Buchmann, Advanced Issues of E-Commerce and Web-Based Information Systems,. WECWIS 2000]

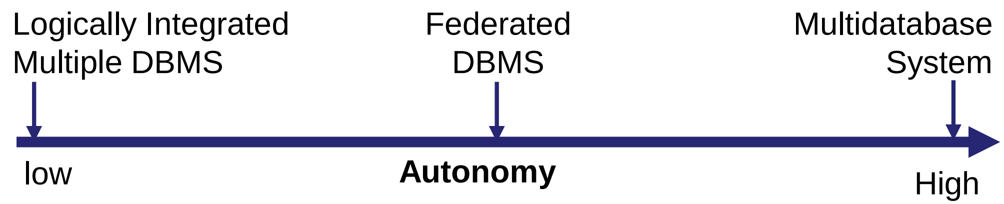
Data Integration - Issues

- Providing a uniform access to multiple heterogeneous information sources
- More than data exchange (e.g., ASCII, EDI, XML)
- Context information is implicit and is lost across system boundaries



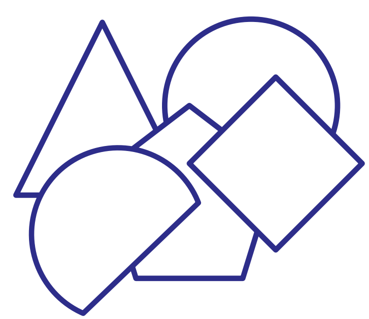
Remember: Problem Space

- Autonomy
  - Transaction Control
  - Query Processing
  - Distribution of Control: Degree of independence of individual DBMS



- Distribution
  - Single DBS
  - Many DBSs in a local network
  - Many DBSs geographically distributed

- Heterogeneity
  - Data Models
  - Schemata and Data Representation
  - Semantics

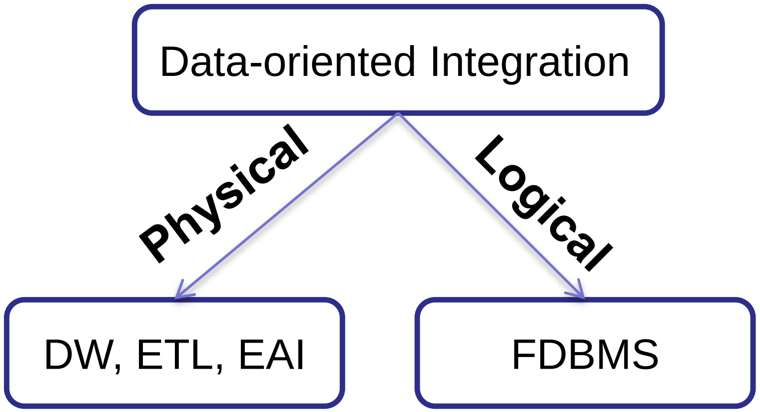


Data Integration

Created with wordle.net based on:
P. Bernstein. Middleware. CACM, Feb. 1996

The many faces of … data integration

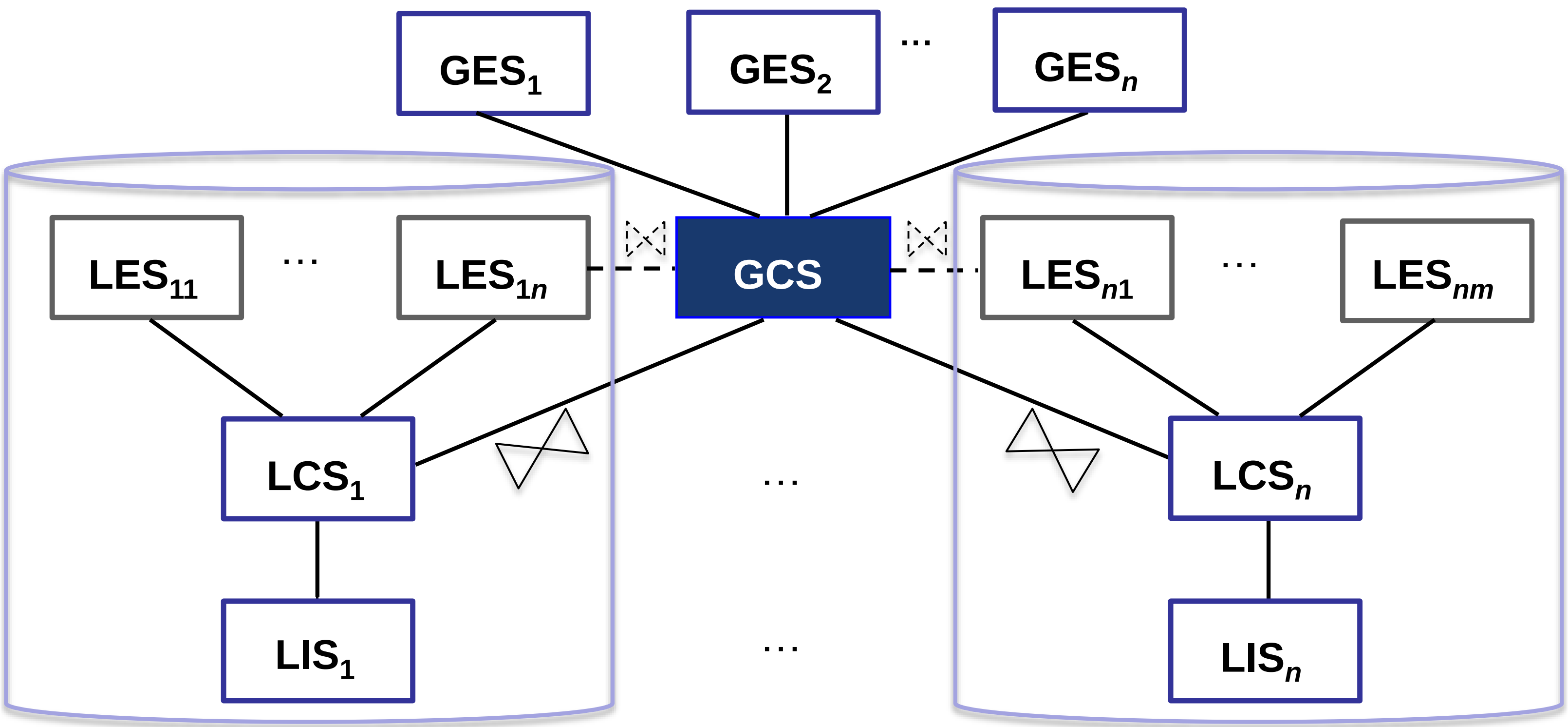
- Problem: Given a set of DBMS (LCS), how to integrate them under GCS and query the data
  - Local Conceptual Schema (LCS), Global Conceptual Schema (GCS)
  - Transformation, Materialization



- Two types of data integration:
  - **Physical** – integrated data is fully materialized
    - Data Warehouses (ETL), Data-oriented EAI
    - Operational vs. Analytical Data stores
    - Materialized View Maintenance
  - **Logical** – schemata are virtually integrated, data partially materialized (query time)
    - EII (Enterprise Information Integration)
    - Federated Databases, Multi-Databases
    - Data maintained in respective local DB
    - Incomplete schema integration → queries decomposed, shipped, evaluated locally

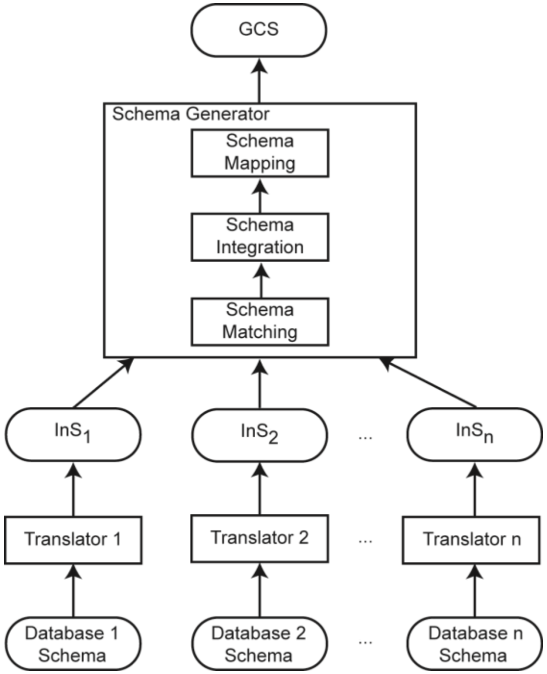
Logical Architecture Multi-DBMS

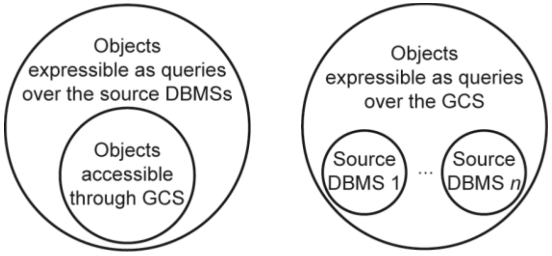
- Global Conceptual Schema != Union of Local DB → U{LCS_i} or U{LES_i}
  - Mediated Schema → Common Data Model
  - Local operations and schemata remain



LCS, GCS, …

- GCS is defined first
  - Derive LCSs from this schema
  - Local-as-view (LAV)
    - The GCS definition assumed to exist, and each LCS is a view definition over it
    - Query Results constrained to LCS definition, although GCS definitions are richer → incomplete answers
- LCS is defined first
  - Global-as-view
    - The GCS is defined as a set of views over the LCSs
    - Query Results constrained to GCS definition, although LCS object definitions are richer
- GCS is defined as an integration of parts of LCSs
  - Generate GCS and map LCSs to this GCS





(a) GAV (b) LAV

Özsu, Valudriez: Principles of Distributed Database Systems

Example LAV

- A simple Example:
- **Source A**: R1(prof, course, university)
- **Source B**: R2(title, prof, course)

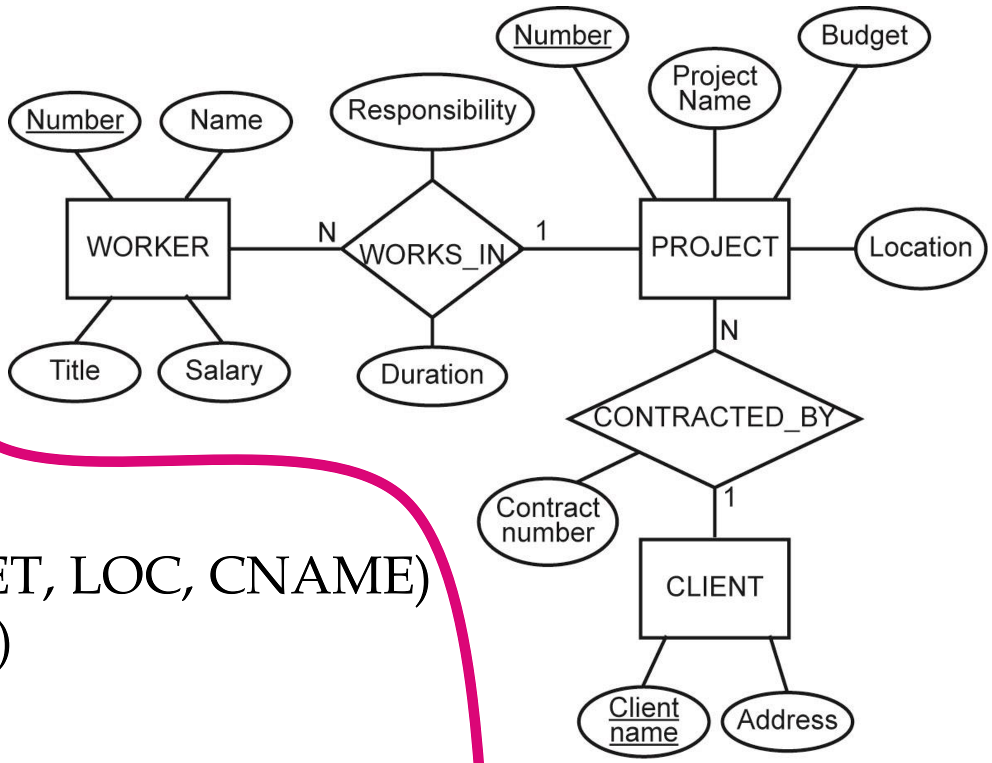
- Definition of the global, integrated schema:
  Global(prof, course, title, university)
- Source A defined as:
  CREATE VIEW R1 AS
  SELECT prof, course, university FROM Global
- Source B defined:
  CREATE VIEW R2 AS
  SELECT title, prof, course FROM Global

Özsu, Valudriez: Principles of Distributed Database Systems

Example

E-R Model

Relational

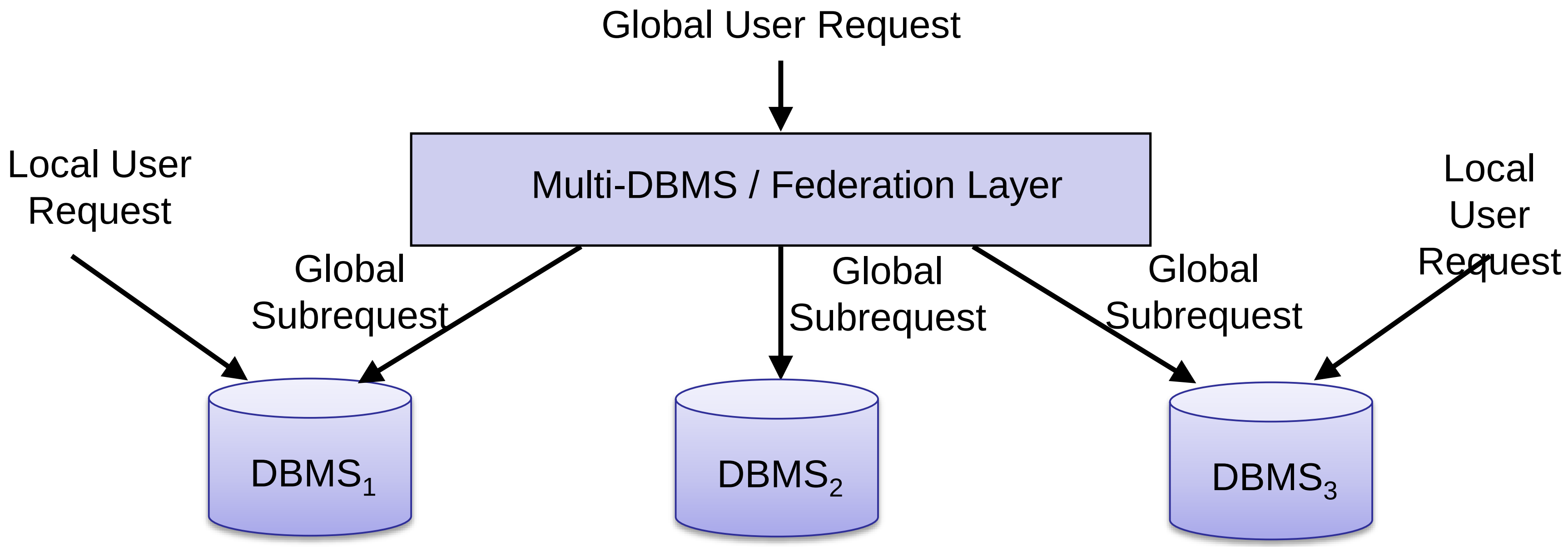


EMP(<u>ENO</u>, ENAME, TITLE)
PROJ(<u>PNO</u>, PNAME, BUDGET, LOC, CNAME)
ASG(<u>ENO, PNO</u>, RESP, DUR)
PAY(<u>TITLE</u>, SAL)

Özsu, Valudriez: Principles of Distributed Database Systems

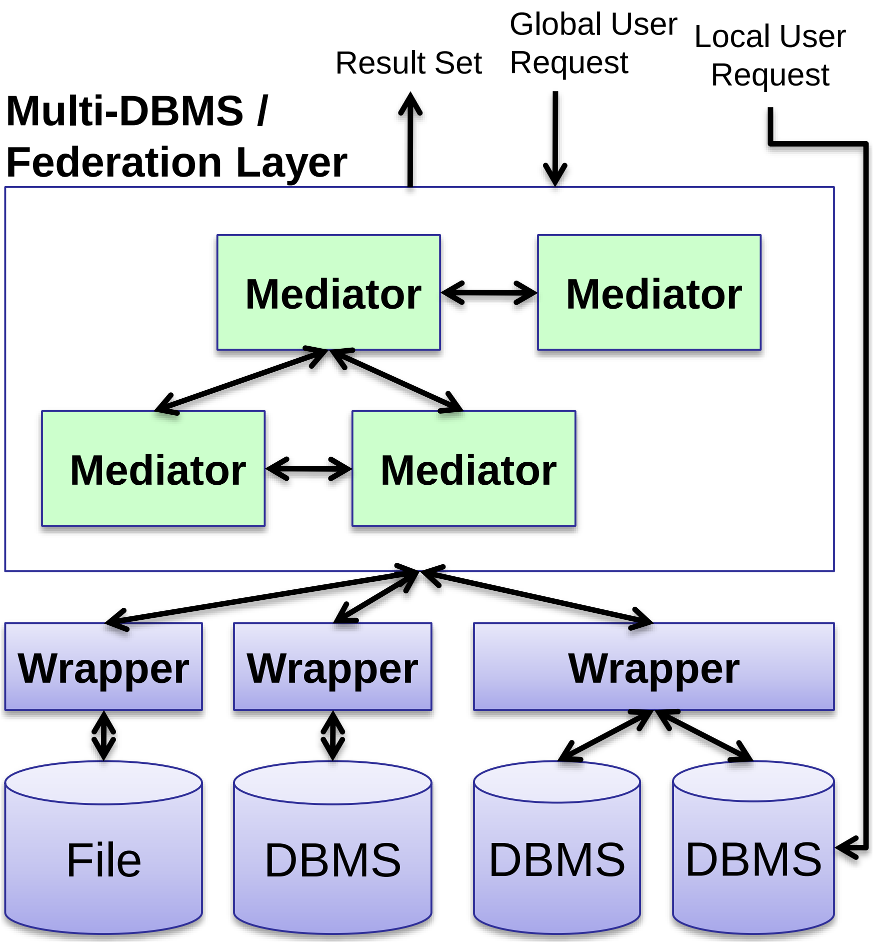
Architectural View

- Transformation: Data, Request/Query
- Data Models
- Query Language(s)



Özsu, Valudriez: Principles of Distributed Database Systems

Implementation of Multi-DB Architecture



- **Mediators** → integrate the data with the same meaning different representations
- **Wrappers** → convert data to a common data model
  - Local Query Processing
  - Standard Interface
- Projects: TSIMMIS, DISCO, Garlic

Özsu, Valudriez: Principles of Distributed Database Systems

Wrapper

- Translate among different data models
  - Data Source Data Model → Canonical Data Model
  - Syntactic heterogeneity
- Data Model
  - Data types
  - Integrity constraints
  - Operations (e.g. query language, insertion, deletion, update)
- Query Processing Capabilities / compensation if missing
- Standard Interface

Mediators

- Integrate data with same "real-world meaning", but different representations
  - Semantic, structural heterogeneity
  - Implement GCS: integration mapping → schema integration

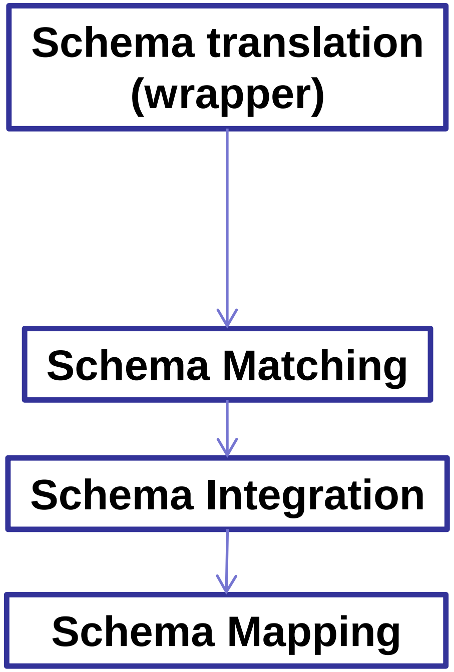
- Decompose queries against the integrated schema to queries against source DBs
  - Decompose Queries: GCS→LCS
  - only for logical integration

Mediator - Schema Integration

- Schema Translation (already discussd)

- Schema Generation – use Intermediate schemata to generate GCS
  - Matching – semantic and syntactic correspondences
  - Integration – integrate common schema elements
  - Mapping - Define mapping functions

- Conflict Resolution → see Heterogeneity
  - Schema level conflicts
    - Naming, Structural, Constraint and behavioral conflicts
  - Data level Conflicts
    - Identification, Representational, Data errors



Mediators

- Common data model → more flexible than common DB Models (ODMG?)

- A mediator model must support
  - A rich collection of structures including nested structures
  - Graceful handling of missing information
  - Meta-information - information about the structures themselves and about the meanings of the terms used in the data

- Common query language:
  - New mediators to join old ones for augmented functionality and
  - New sources to provide input to an existing mediator

- Tools to automate the creation of new mediators and mediator systems
  - Generators
  - Declarative specification

Garcia-Milina, Ullman et. al The TSIMMIS Approach to Mediation: Data Models and Languages

Issues with the architecture

- Mediator Stability
  - Mediator schema may have to be changed when a new source is added

- Mismatch among the capabilities of data sources
  - Different wrappers may have different functionality

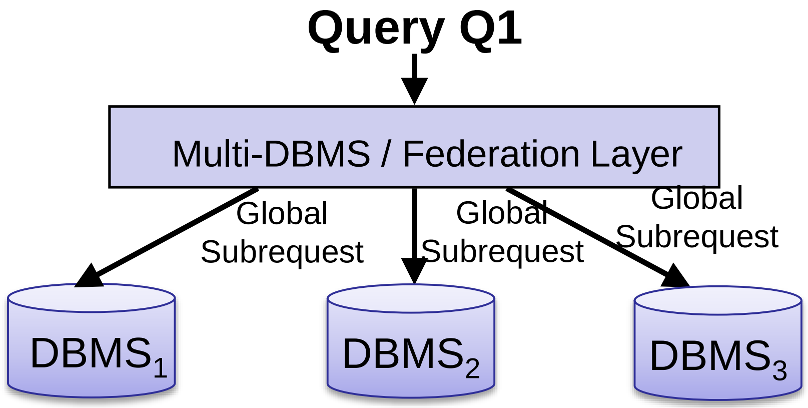
- Low Fault Tolerance
  - Queries can not be processed if a data source is unavailable

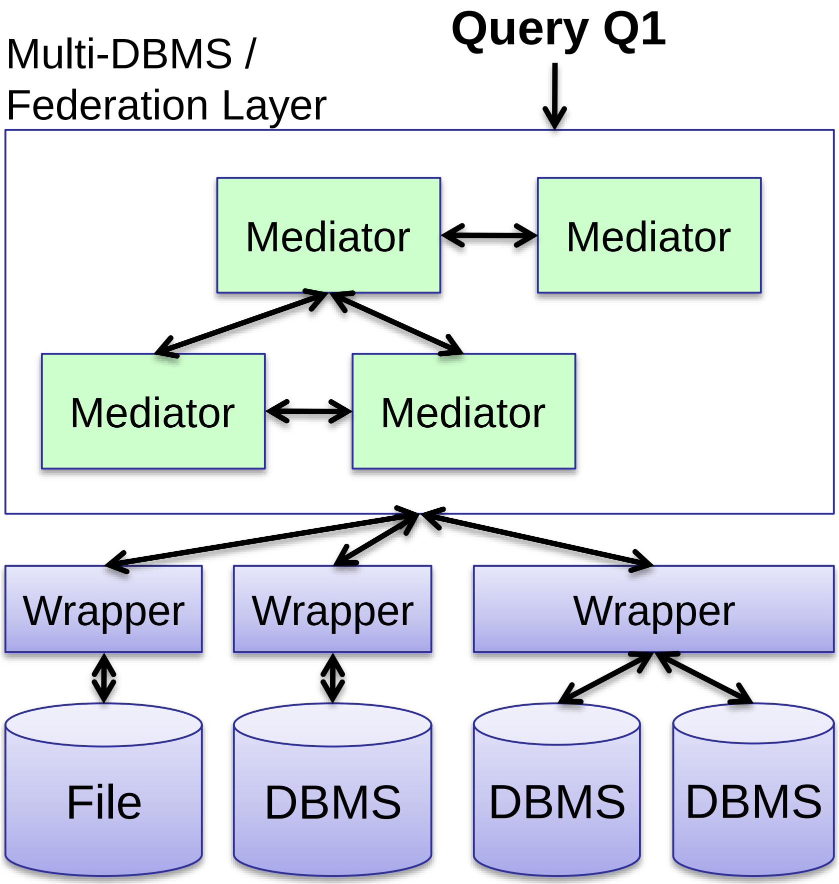
Query Processing

Created with wordle.net based on:
P. Bernstein. Middleware. CACM, Feb. 1996

Query Processing – Big Picture

- Execute a query on 'virtually' federated data under problem space constr.
- Execute parts locally at data store - compensate for the rest
  - Minimize transferred data volume
  - Reduce Roundtrips → Latency-aware
  - Evaluate close to data
- Query Decomposition
- Capability Negotiation





Özsu, Valudriez: Principles of Distributed Database Systems

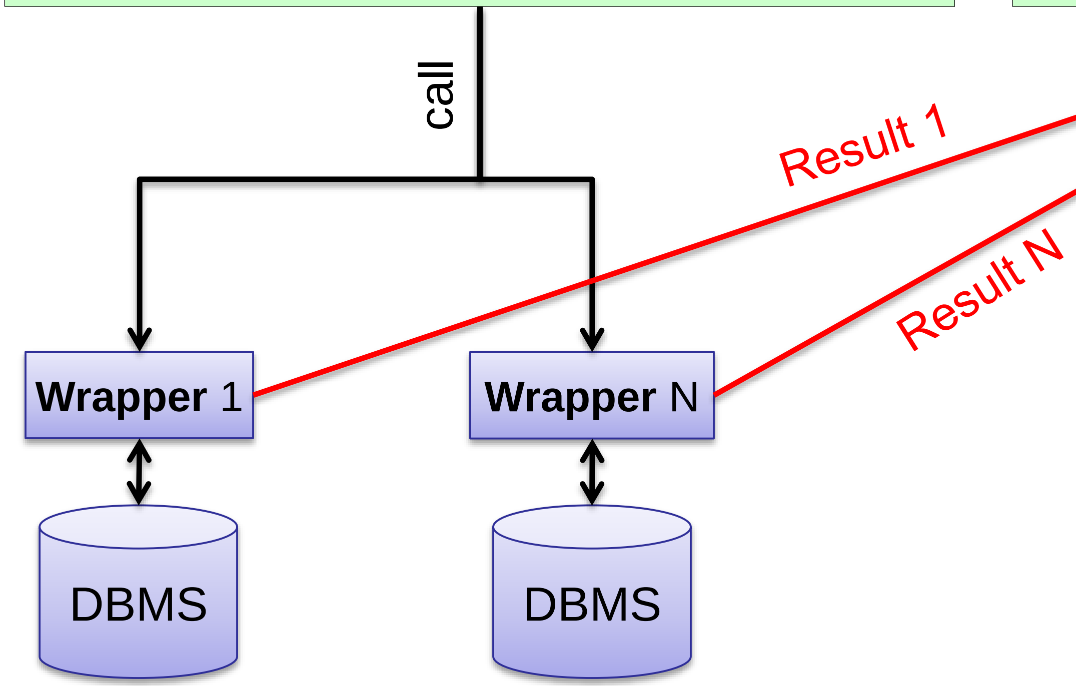
Mediator Query Processing

Mediator Query Processor

1. Reformulate the Query for Local Schema
2. Build Logical Operator Tree
3. Identify Sub Trees Executable by Wrappers
4. Generate Distributed Execution Plan

Mediator Run-Time

5. Execute Composition Query



Query Processing

- Reformulate the query → for local schemas
- Transform the query into logical operator trees
- Decompose query into
  - wrapper sub-queries and
  - a composition query
- Adapt the wrapper sub-queries and the composition query to reflect the capabilities of the wrappers
- Generate distributed execution plans
- Estimate the minimum cost plan
- Send the wrapper sub-queries to the wrappers
  - execute the composition query on the results

Query Processing

- Optimization strategy: push parts of the plan to wrapper | Global plan opt.

- Wrapper schema: schema entities, sizes

- Wrapper Capabilities
  - Wrapper exports information about operators executable on schema entities
  - SELECT [on ABC] … PROJECT on [XYZ] … SCAN [on ALL]

- The Mediator has a generic cost model
    - sequential scan and index scan
    - index join, nested loops and sort-merge join
    - index existence
    - cost functions (derived through calibrating)

- Wrapper can override the mediator model
  - exporting statistics
  - cost functions

Summary

- System Federation
- Enterprise Application Integration

- Problem Space

- Data-Oriented EAI
  - Physical, Logical

- Logical Data Oriented Integration
  - Schema Integration
  - Wrapper/Mediator Approach
  - Federated Systems

Thank You!

TECHNISCHE UNIVERSITÄT DARMSTADT

Questions?