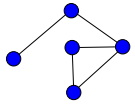
Gliederung

Peer-to-peer Systeme und Datenbanken(SS07)

- Kapitel 1: Einführung
- Kapitel 2: Beispiele
- Kapitel 3: Routing
- Kapitel 4: Schemabasierte p2p-Netzwerke
- Kapitel 5: Integrationsprobleme
  Teil 1:
  Teil 2:
  Teil 3:
- Kapitel 6: Anonymität, Authentifikation
- Kapitel 7: Dienstgüte

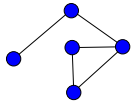
Version vom 16. Mai 2007



Kapitel 4

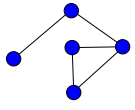
Schemabasierte P2P-Netzwerke

- Abgrenzung zu bisher besprochenen Systemen
- Anforderungen
- Techn. Hilfsmittel:RDF, OIL
- Integration
- Verteilte Queries
- Beispiel: PIER



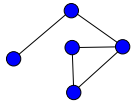
Anforderungen für P2P-DB:

- komplexes Schema, d.h.
  Definition: *komplex*
  es ex. eine Schemabeschreibungssprache, mit der Schemata definiert und erweitert werden können.
  $\mapsto$ verschiedene Knoten mit evt. versch. Schemata bzw. Sichten auf Daten - Heterogenität.
  $\mapsto$ im Schema Metainformationen über Daten (Autor, Gebiete, ..) in starkem Maße vorhanden und Gegenstand von *komplexen* Anfragen (incl. Anfragen nach Metadaten).
  $\mapsto$ verschiedene Operatoren, z.B. JOIN, Gruppierung, Aggregate-, Sortierfunktionen über mehrere Knoten (ggf. im gesamten Netz)
- Mehrschichtige Architektur, meist basierend auf Hash-Tabellen und Routingindices u.ä.; Mechanismen zur Anfrageweiterleitung, darüber Anfragebearbeitung und Nutzerinterface.
- aus P2P: Kein globales Wissen, Knotenautonomie, fluktuierende Teilnehmer und Datenbestände



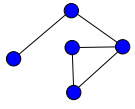
Grundprobleme

- Grundlegende Diskrepanz zwischen Transaktionseigenschaften einer DB und der Autonomie der Peers.
  Vergleichbar: Föderierte Datenbanksysteme mit Fluktuation und Autonomie der Komponenten
  Semantische Integrität i.a. nicht gesichert.
- Datenintegrationsprobleme (wie aus FDBS bekannt)
  Lösung 1: Wrapper, Mediatoren, Exportschemata
  bedingt geeignet, da globales Schema unterstellt wird.
  Lösung 2: Semistrukturierte Daten, z.B. XML
  Lit.: Risse, T. und Knesevic,P.: *A Peer-to-Peer XML Database.* Fraunhofer IPSI. 2003.
  Integration auf Instanz- oder Schemaebene.
- Ressourcenallocation: ggf. komplexer, da Metadaten mit verwaltet.
  Nutzung von Hash-Funktionen: Ausrechnen, welcher/welche Server welche Resource verwalten (ähnlich CHORD)
- Anfragebearbeitung: Zerlegung einer Anfrage in Teile für einzelne Peers, Zusammenfügen (UNION, JOIN) der Teilantworten.



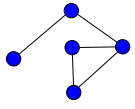
Gegenüberstellung

	P2P (2007)	?	FDBMS
Granularität	Vollständige Dateien		Objekte, Tupel, Attribute
Anfragen	Simpel: Dateinamen, Hashwerte		Komplex: SQL, XQuery
Antworten	meist unvollständig		vollständig, exakt
Update, Version	meist mangelhaft		DB-artig
Schema	höchstens Techn. Tab.		Exportschema
Systemkenntnis	kein glob. Wissen		Komponenten bek.
Anfrage-bearbeitung	Fluten, Hash, part. Kenntnis (Nachbarn)		Optimierend, 2PC
Dynamik	Persistenzproblem		Verfügbarkeit wie DBMS
Kardinalität	sehr groß		wenige DBVS



Gegenüberstellung (Fortsetzung)

	P2P (2007)	P2P-DB	FDBMS
Granul.	Vollst. Dateien	Obj., Tub., Attr.	Obj., Tub., Attr.
Anfrage	Simpel: Namen, Hashwerte	Komplex: SQL, XQuery, (?) ...	SQL, XQuery,...
Antwort	meist unvoll.	vollst. (?), exakt Qual.-Bewert.	v., e.
Versionen	meist mangelh.	DB-artig	DB-artig
Schema	Techn. Tab.	paarw.vorberechn. Integr. on the fly ?	Exp.-schema.
Sys-kenntn.	kein glob. Wissen	mind. Nachbarn glob. Struktur ?	Komponenten bekannt
Anfrage-bearb.	Fluten, Hash, part. Kenntnis (Nachbarn)	Mappings, Hash Serverfunkt. Koord. 2PC o.ä.	Optimierend, 2PC
Dynamik	keine Persistenz	Persistenz (?)	DBMS
Kardinalität	sehr groß	mittel (derzeit)	wenige DBVS



Technische Hilfen: RDF

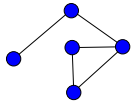
RDF *Resource Description Framework* (*http://www.w3.org/RDF/*)
ermöglicht die Beschreibung von WEB-Resourcen durch Metadaten,
automatisch verarbeitbar.

- Datenmodell: Beschreibung einer Ressource durch *properties*:

$$< property, subject, value >$$

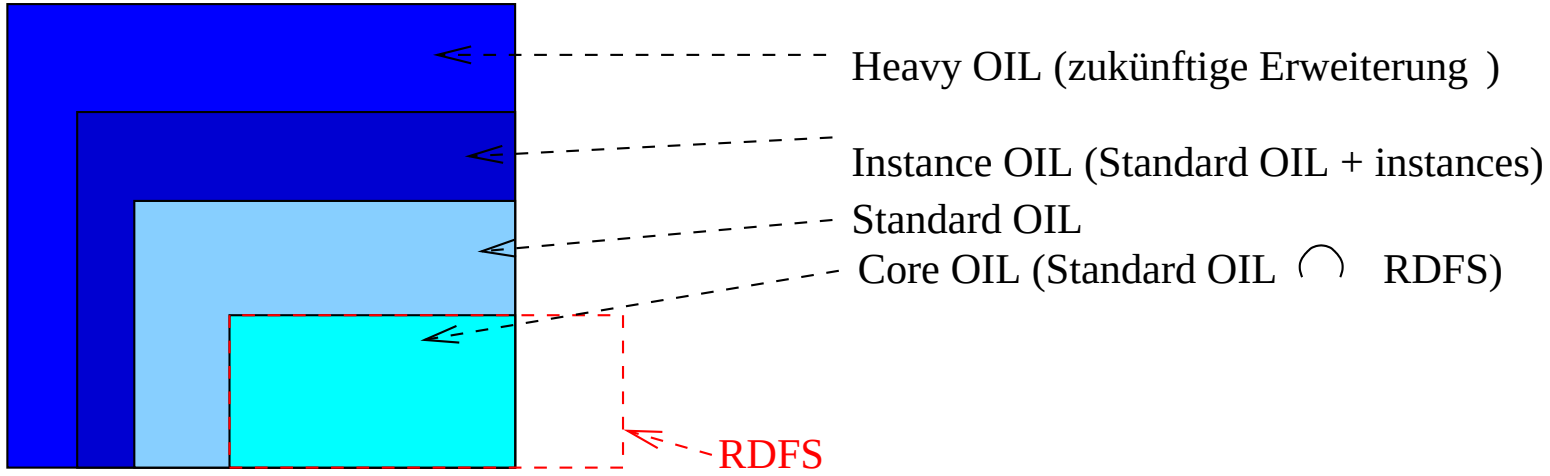
$subject$: die zu beschreibende Resource
$property$: die zum Subjekt gehörige Eigenschaft
$value$: der Wert der Eigenschaft.

- Beispiel:
$\{Autor, [http://www.informatik.uni-leipzig.de/\sim sosna], DieterSosna\}$
in XML:
<rdf:Description about='http://www.informatik.uni-leipzig.de/~sosna'>
<autor>Dieter Sosna</autor>
</rdf:Description>



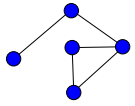
Technische Hilfen (2): OIL

OIL *Ontology Interchange Language*, *http://www.ontoknowledge.org/oil/* ermöglicht formale Definitionen für Ontologien



Schichtenarchitektur, abwärtskompatibel: Automaten, die untere Schichten verstehen, verstehen auch Teile aus den oberen Schichten.

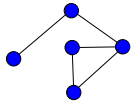
- OIL-Definition besteht aus Klassen und Beziehungen. Klassen durch Eigenschaften(Attribute) beschrieben, Beziehungen = Verhalten der Instanzen.
- Core Oil: kann von RDF-Schema Agenten verarbeitet werden
  Instance OIL: Integration über Datenbankfähigkeiten.



Integration auf Instanzebene

Local Relational Model (LRM)
Lit.: Bernstein,A. Ph. u.a.: *Data Management for Peer-to-Peer Computing: A Vision*. http://www.db.ucsd.edu/webdb2002/papers/15.pdf

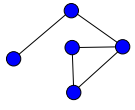
- Architektur:
  auf lokalem DBVS in allen Knoten eine identische Schnittstelle (LPM-Ebene) mit User-Interface (UI), Query-Manager (QM), Update-Manager (UM) und Wrapper(zwischen UM,QM und der lokalen DB).
- Bekanntschaften, Koordinierungsformeln:
  2 Peers haben Bekanntschaft, wenn es Koordinierungsformeln gibt, die semantische Beziehungen / Abhängigkeiten zwischen deren Daten beschreiben, die zeigen, wie sich die eigenen Daten für den Bekannten darstellen, wie die Elemente der einen DB in Elemente der anderen DB übersetz werden müssen (Binäre Domainbeziehungen).
- Regeln formal beschrieben.



Integration auf Instanzebene (2)

Lit.: Arenas, M. u.a.: *The Hyperion Project: From Data Integration to Data Coordination*. ACM SIGMOD Record, 32(3), 2003

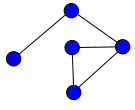
- Architektur ähnlich LRM, statt UM jetzt Rulemanager (RM), der Konsistenzregeln, die über Nutzerschnittstelle eingegeben werden, durchsetzt
- Mappingtabellen definieren die Überführung von Werten zwischen je zwei DB.
- Das Aufstellen der Mappingtabellen erfordert Kooperation der Partner, geschieht mit Nutzerinteraktion.



Integration auf Schemaebene

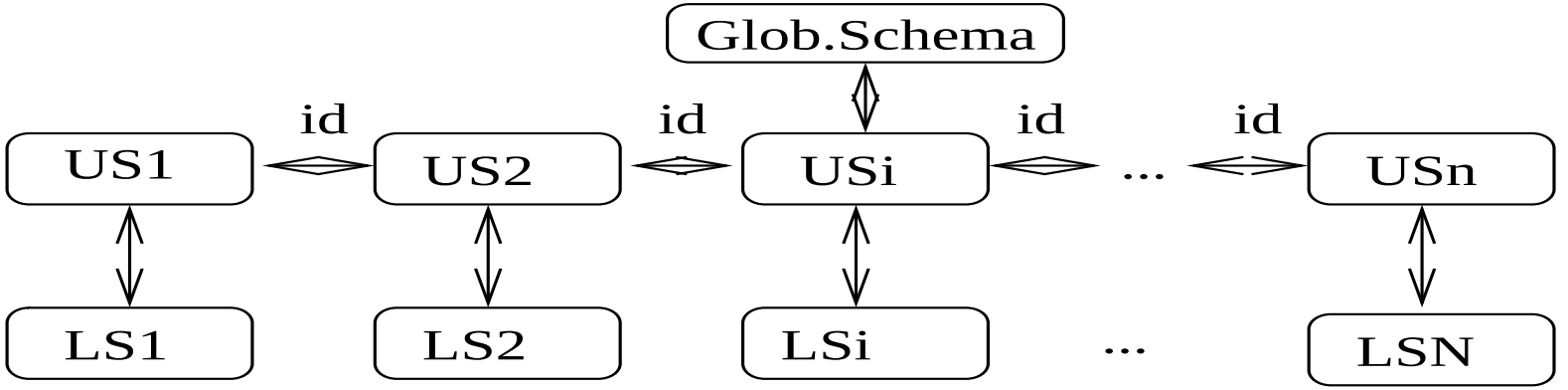
AutoMed: Automatic Generation of Mediator Tools for Heterogeneous Database Integration
Lit.: Poulovassilis, A.: *AutoMed: ....*
http://www.dcs.bbk.ac.uk/ ap/talks/abdn2003AutoMrdPres.ppt

- Zwei Transformationsrichtungen:
  (1) GAV - global-as-view: Die Relationen des vermittelten (globalen) Schemas werden als Menge von Sichten der Relationen der Datenquelle beschrieben.
  (2) LAV - local-as-view: Die Relationen der (lokalen) Datenquelle werden als Sichten auf dem (globalen) vermittelten Schema dargestellt.
  (3) BAV - both-as-view: Kombination von LAV und GAV.
- Superpeer liefert ein globales Schema.



AutoMed (2)

Schematransformation: (US_i union-compatible Schemata)



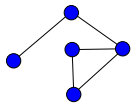
- Grundoperationen: add, delete, rename, expand, contract; aus denen alle Umformungsregeln zusammengesetzt. Alle Modellierkonstrukte in Hypergraph Data Model (HDM) Termen dargestellt, einheitliche Beschreibungssprache.
- Model Definition Repository (MDR) -

Berkley peer schema (XML DTD):

Elemant schedule(college*)
Element college(name, dept*)
Element dept(name, course*)
Elemant course(title, size)

MIT peer schema:

Element catalogue(course*)
Element course(name, subject
Element subject(title,enrollme



Integration auf Schemaebene Piazza

Lit.: Halevy,A. u.a.: *Piazza: Data Management Infrasructure for Semantic Web Applications*.
http://www.cis.upenn.edu/ zives/rearch/piazza-www03.pdf

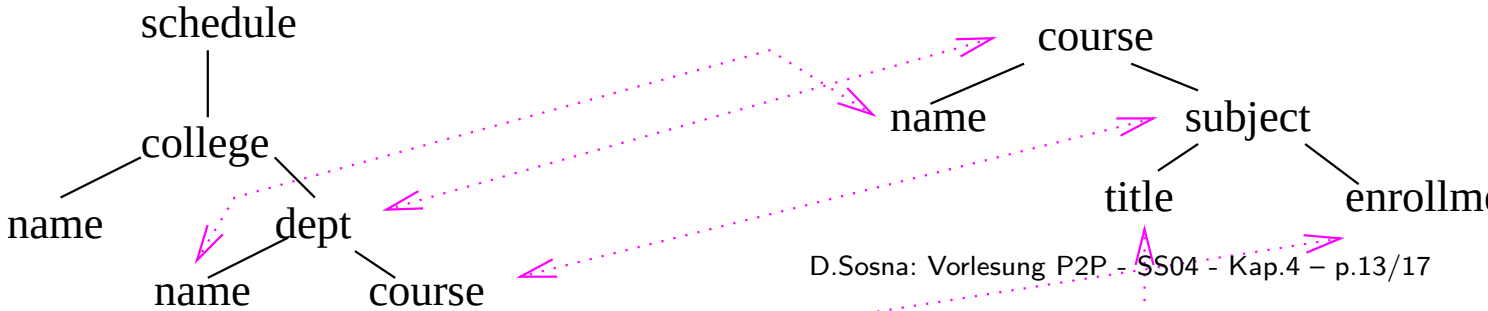
- Piazza nutzt transitive Beziehungen zwischen den Schemata der beteiligten Peers.
- Kooperative Peers vorausgesetzt, die paarweise Mappings ihrer Schemata definieren. Eingebrachte Resourcen:
  (1) Daten (XML-, RDF-Dateninstanzen) (2) Metadaten (XML-Schema,

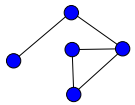
Berkley peer schema (XML DTD):

Elemant schedule(college*)
Element college(name, dept*)
Element dept(name, course*)
Elemant course(title, size)

MIT peer schema:

Element catalogue(course*)
Element course(name, subject*
Element subject(title,enrollme

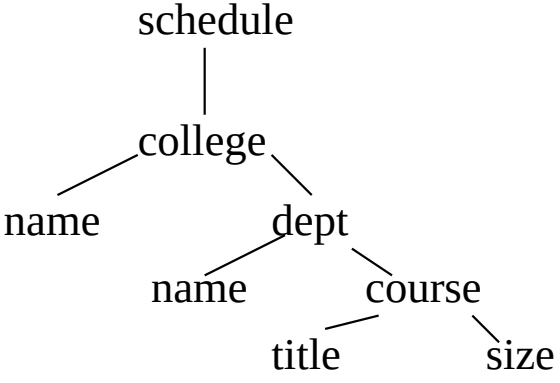




Piazza Beispiel

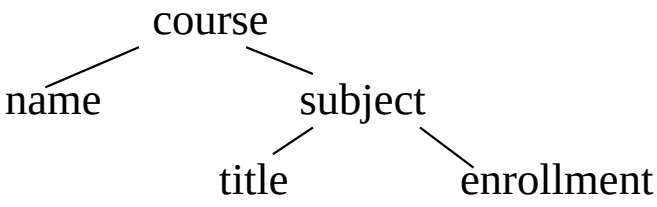
Berkley peer schema (XML DTD):

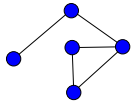
Elemant schedule(college*)
Element college(name, dept*)
Element dept(name, course*)
Elemant course(title, size)



MIT peer schema:

Element catalogue(course*)
Element course(name, subject*)
Element subject(title,enrollment)





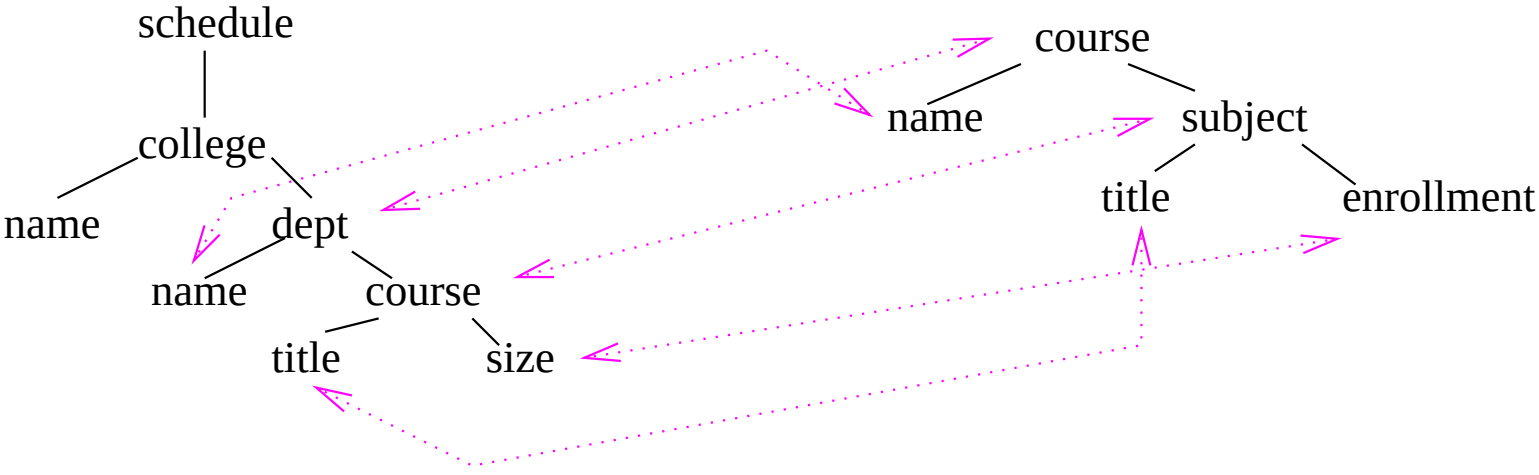
Piazza Beispiel (2)

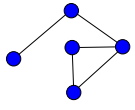
Berkley peer schema (XML DTD):

Elemant schedule(college*)
Element college(name, dept*)
Element dept(name, course*)
Elemant course(title, size)

MIT peer schema:

Element catalogue(course*)
Element course(name, subject*)
Element subject(title,enrollment)





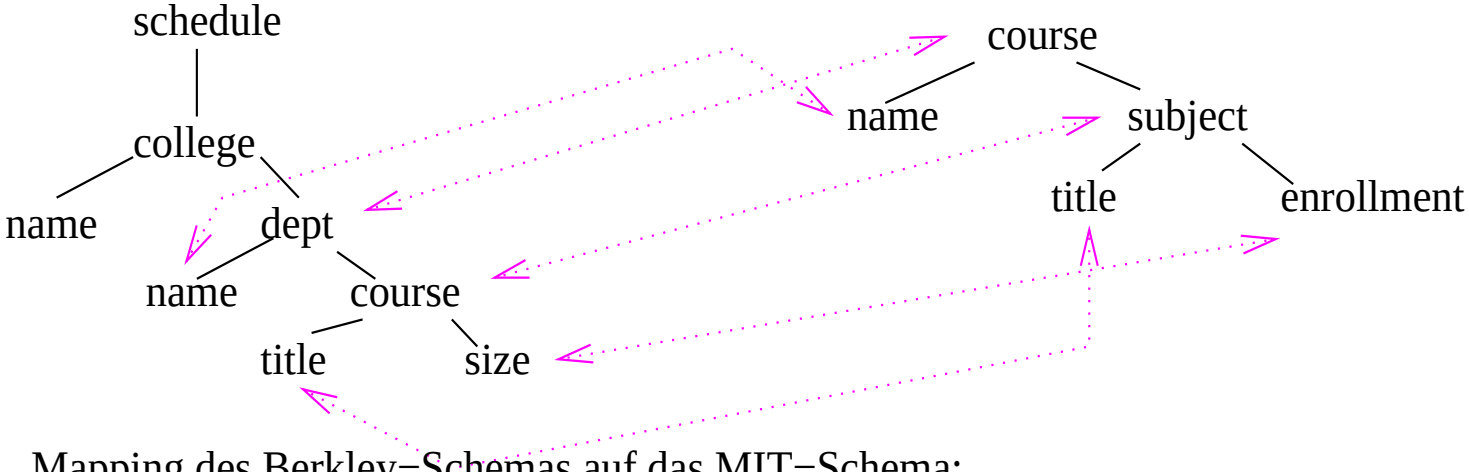
Piazza Beispiel (3)

Berkley peer schema (XML DTD):

Elemant schedule(college*)
Element college(name, dept*)
Element dept(name, course*)
Elemant course(title, size)

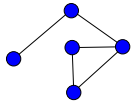
MIT peer schema:

Element catalogue(course*)
Element course(name, subject*)
Element subject(title,enrollment)



Mapping des Berkley−Schemas auf das MIT−Schema:

```
<catalog>
    <course> {$c=document("Berkley.xml")")/schedule/college/dept}
        <name> $c/name/text() </name>
        <subject> {$s = $c/course}
            <title> $s/title/text() </title>
            <enrollment> $s/size/text() </enrollment>     </subject></course></catalog>
```



Zusammenfassung

- 2007: Konzepte und Realisierungen von P2P und DBS stark unterschiedlich
- Zusammenführung erfordert (z.T. neue) Lösungen für
  - Schema- und Datenintegration, -transformation (automatisiert, große Schemata, kurzfristig - „Echtzeit“).
  - Anfrageverteilung, Anfrageoptimierung
  - Adäquates Transaktionskonzept (P2P-Variante des 2PC ?)
  - Qualitätsbewertung: Akzeptiert der Peer DB-Merkmale ?
- Kompromisse
  - Die letzten 3 Punkte - klare Merkmale eines DBVS.
  - Die letzten 2 Punkte - Autonomieeinschränkung der Peers.
  - Einschränkungen von DB-Merkmalen ? (Vollständigkeit d. Ergebnisse ? Wiederholbarkeit einer Anfrage ?)