Universität Leipzig
Institut für Informatik
Abteilung Datenbanken
Problemseminar: P2P Data Management
Seminarleiter: Prof. Dr. Erhard Rahm
Betreuer: Hong Hai Do
WS 2003/04

Schriftliche Ausarbeitung des Referatthemas:

Datenintegration in P2P-Datenverbünden

Mohammad Esad-Djou
bss98aou@studserv.uni-leipzig.de
HF Informatik NF Kommunikations- & Medienwissenschaft (9.FS)
Matrikel-Nummer: 8121673

Inhaltverzeichnis

1. Einführung

Das Konzept des Peer-to-Peer (im Folgenden P2P)-Computing basiert vor allem auf einem direkten und dynamischen Knoten-zu-Knoten-Modell der Kommunikation. Peers wirken sowohl als Clients als auch als Server. Sie sind auch autonom in Bezug auf die Steuerung und die Strukturierung des Netzes; neue Peers können das Netz jederzeit betreten oder verlassen. Das P2P-Computing-Paradigma ist durch den Mangel an globaler Steuerung in Form von globalen Registrierungen, globaler Anwendungen, globaler Ressourcenverwaltung und eines globalen Schemas und Datenlagers charakterisiert. [Hy04]

Obwohl das P2P-Paradigma immer öfter Anwendung findet, behandeln P2P-Anwendungen im Allgemeinen das gemeinsame Objekt, z.B. eine Datei als Einheit, und verwalten seinen Inhalt nicht. Systeme wie *Napster* [Na04] und *Gnutella* popularisierten das P2P-Paradigma als eine Version traditioneller verteilter Systeme. In der Tat basieren viele bewährte Systeme auf ähnlichen Grundlagen.

Napster ist ein zentralisiertes P2P-System: Der Napster-Server benutzt eine zentrale Tabelle, in der die derzeit teilnehmenden Peers verwaltet werden. Diese Tabelle bzw. dieser Index besteht aus dem jeweiligen Dateinamen und der Peer-Adresse (user, files) [AH02]. Daten werden in einzelnen Peers verwaltet. Der Server verwaltet einen ständig aktuellen Index von den Peers, die die jeweils angeforderte Datei gerade anbieten. Jeder Peer, der das Netz betritt, sendet die Namen der Dateien, die er zur Verfügung stellt an den Server. *Gnutella* [Gn04] und *Freenet* sind dezentralisierte File-Sharing-Systeme. Über das Gnutella-Protokoll können Teilnehmer eines virtuellen Netzes in einer Art P2P kommunizieren, indem sie ein einfaches Protokoll für verteilte Dateisuche nutzen. Freenet füllt durch Interaktion mit dem Netz seine anfangs leere Tabelle aus. Die Routing-Tabelle bleibt in den Servents von Freenet und speichert die *Adressen* der Nachbarpeers sowie die dort gespeicherten *Schlüssel* von Datenelementen und eine Liste der entsprechenden *Daten* (ein drei spaltige Routing-Tabelle aus: *Key, Data, Address*).

Wenn Architektur und Datenintegration dieser Systeme genauer betrachtet werden, wird deutlich, dass das Datenmanagement dieser P2P-Systeme auf die Datei-Ebene reduziert ist. Diese Systeme sind einfach Datenaustauschprogramme, die keinen gemeinsamen Ansatz für eine inhaltliche Untersuchung der "Menge der Daten" vorgesehen haben. Sie erzeugen auf verschiedene Art und Weise eine Tabelle von Peer-Adressen, Dateinamen usw.

1.1 Motivation und Problemstellung

In der Vergangenheit tendierten einzelne Forscher dazu, Daten einzeln zu erfassen und zu analysieren, Small-Scale-Phänomene zu studieren und die erfassten Daten als ihr Eigentum zu betrachten. Heute besteht ein großes Interesse daran, die erzeugten Daten zusammenzufassen, um ein globales Bild zu bekommen. Wenn die Interoperabilität und Datenintegration zwischen heterogenen Datenquellen ohne Probleme möglich wäre, könnten dann alle anderen Forscher freien Zugang dazu haben. Heute besteht ein viel größerer Bedarf an Informationszugang und Informationsaustausch. Dementsprechend müssen sich Techniken und Methoden zur Lösung dieser Aufgaben entwickeln.

Als ein Motivationsbeispiel wird hier die biologische Forschung betrachtet. Gegenwärtig gibt es eine überwältigende Anzahl von Genomic-Datenquellen, die sich von großen öffentlichen Quellen wie GDB [Ge04] bis hin zu Quellen erstrecken, die auf einzelne Forschungslaboratorien beschränkt sind. Die Integration dieser Quellen ist wichtig, um Wissenschaftlern einen einheitlichen Zugriff auf alle relevanten Daten zu ermöglichen.

Wegen einer Unzahl von politischen, finanziellen und technischen Gründen scheint dieses Ziel jedoch unerreichbar. Zu den technischen Gründen gehört die inhärente Heterogenität von Quellen, die sich von relationalen Datenbanken bis zu formatierten Dateien oder Spreadsheets erstrecken.

Hier stellt sich die Frage, wie verschiedene Datenbanken Informationen austauschen und ihre Informationen immer auf dem aktuellsten Stand halten können. Wie sollen diese Datenbanken gegenseitig ihre Schemata vergleichen, um gegebenenfalls ihre Schemata kompatibel zu machen? Wie erfolgt der Zugriff auf Daten und/oder die Änderung der Datenbanken in P2P-Systemen? Haben die Daten in verschiedenen Datenbanken gemeinsame Datenmodelle und wenn ja, welche? Wenn nicht, wie kann man verschiedene Datenmodelle transformieren?

Um dieses Beispiel zu präzisieren, kann man die Gen-DB (GDB), die SwissProt-DB [Sw04] und MIM-DB [On04] betrachten. Diese Datenbanken sind autonom von verschiedenen Organisationen erstellt. Der Bedarf für Datenaustausch zwischen biologischen Datenbanken ist vorhanden, aber zahlreiche unterschiedliche Formate und Schnittstellen verhindern oder beschränken eine effektive Zusammenarbeit zwischen Forschungszentren und eben ihren biologischen Datenbanken. Z.B. müssen Daten von Gen-Identifiern aus GDB, Protein-Identifiern aus SwissProt, Krankheitsgenen aus MIM und DNA-Segmenten aus MIM und GDB miteinander verglichen und/oder deren Datenbanken zusammengeführt werden können. Ferner ist ein Zugriff von verschiedenen Forschern oder Forschungszentren in einer P2P-Umgebung mit heterogenen Datenstrukturen notwendig.

Abbildung 1 zeigt den Zusammenhang dieser Forschungszentren als Peer A, B und C. Parallel sieht man gegenseitige Verbindungen zwischen drei Datenbanken und einzelner Forschungszentren in einer P2P-Umgebung.

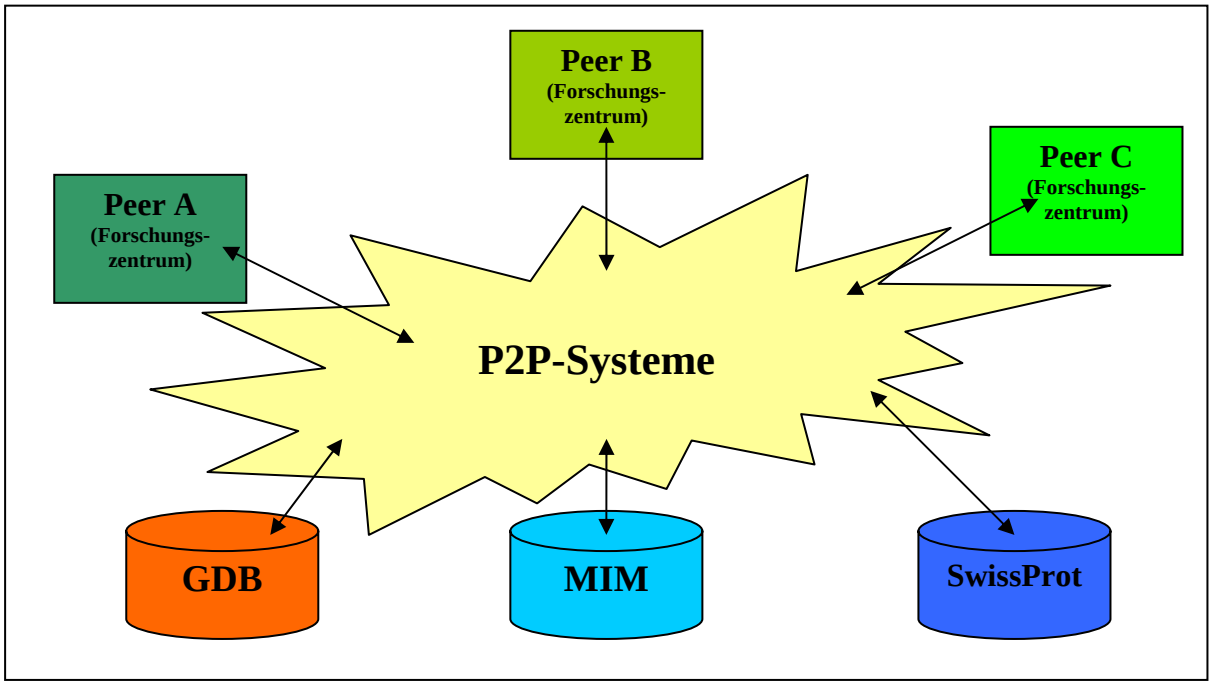

Abbildung 1. Wie können Daten in drei DB GDB, MIM und SwissProt in eine gemeinsame P2P-Umgebung integriert werden?

Kurz gesagt, die gemeinsame Nutzung der semantisch unterschiedlichen Daten mit verschiedenen Schemata ist inhärent ein schwereres Problem. Probleme der Integration verschiedener Informationsquellen sind sowohl semantischen Ursprungs, z.B. [BK+03]:

a) *Heterogenität*

b) *Diversivität der Anfragesprachen:* es existiert im Bereich P2P-basierter Informationsquellen bisher kein standardisierte Anfragesprache wie beispielsweise SQL.

c) *Verwendung kontextspezifischer Metadatenstandards:* viele Informationsquellen beschreiben ihre Ressourcen über Metadaten. Es werden unterschiedliche Standards zur semantischen Beschreibung verwendet.

d) *Fehlende Metadaten:* steht in Datenbanken grundsätzlich das Schema als beschreibende Information zur Verfügung, fehlt bei P2P-Quellen häufig eine nähere Beschreibung zur Semantik und Struktur der bereitgestellten Daten.

als auch technischer Natur, wie:

a) *stark dynamische Infrastrukturen:* aufgrund der hohen Autonomie des P2P-Systems ist die Frage der Evolutionsfähigkeit bei der Integration von P2P-Quellen verstärkt zu betrachten. Dies gilt sowohl für Änderungen der Gesamtinfrastruktur durch Hinzufügung oder Wegnahme von Quellen und für technische Änderungen der Schnittstellen als auch für Änderungen der Inhalte und deren Beschreibung.

b) *unterschiedliche Schnittstellen und Protokolle*

Für eine effektive Datenintegration in P2P-Systeme muss man diese Probleme lösen. Es sind verschiedene Lösungsansätze für die Datenintegration in P2P-Systeme vorgeschlagen worden. In dieser Arbeit wird auf einige von ihnen eingegangen.

1.2 Ziel dieser Arbeit

Die Mängel der existierenden P2P-Systeme und die Unterschiede in den Eigenschaften der Datenintegration zwischen traditionellen Ansätzen (s. u.) und P2P-Systemen führen dazu, dass die Notwendigkeit der Beachtung der P2P-Datenintegration höhere Priorität bekommt. Das Interesse dieser Arbeit besteht darin, folgende Aspekte, die für Datenintegration und Beseitigung semantischer Unvollständigkeit in P2P-Systeme entscheidend sind, genauer zu beschreiben:

- *Datenmodelle:* Wie werden Daten repräsentiert bzw. integriert? Wie werden die Daten modelliert?
- *Datenintegration auf Instanz-* oder *Schema-Ebene*: Es ist zu beachten, dass die Integration auf Datei-Ebene, Instanz-Ebene oder Schema-Ebene erfolgen kann. Für jede Ebene ist eine konkrete Lösung erforderlich. Wenn ein Datenschema für die verschiedenen Peers vorausgesetzt wird, dann müssen Daten in verschiedenen Schemata irgendwie abgebildet oder transformiert werden.
- *Zentralisiertes* oder *dezentralisiertes Modell:* Wie findet ein Peer relevante Peer-Daten bzw. Datenbanken? Der Lösungsansatz benötigt eine zentralisierte oder dezentralisierte Architektur.
- *Datenaustausch:* Wie werden Dateien oder Daten gesucht und zwischen Peers ausgetauscht? Wie wird eine Anfrage gestellt? Das Objekt eines Datenaustausches kann aus einer Datei, einer semi-strukturierten Datei oder einer DB bestehen. Die Anfrage kann eine einfache Dateisuche oder eine Anfrage an eine DB sein. Im ersten

Fall wird ein String innerhalb von einem Index gesucht, im zweiten Fall wird ein gemeinsames Schema oder eine dialogfähige (interoperable) Peer-DB vorausgesetzt.

Es wird versucht einen kurzen Überblick über Forschungsansätze für die Lösung der Problematik Datenintegration in P2P-Systeme zu geben.

Die Arbeit ist wie folgt gegliedert: Zuerst werden in Kapitel 2 die traditionellen Datenintegrationsansätze genannt und ihre Beschränkungen in P2P-Systeme erwähnt. In Kapitel 3 werden ausgewählte Forschungsansätze vorgestellt, und im 4. Kapitel werden diese miteinander verglichen.

2. Traditionelle Datenintegrationsansätze und P2P-Systeme

Ein Problem der P2P-Anwendungen besteht darin, dass Daten von verschiedenen Quellen keine gemeinsame bzw. standardisierte Struktur, Typ und Einheit haben. Datenquellen sind heterogen, z.B. relationale DB, Webanwendungen, XML-DB, Dateisysteme usw. In zwei verschiedenen Peers können die gleichen Daten unter verschiedenen Namen abgelegt sein, die gleichen Attributnamen für verschiedene Zwecke benutzt werden oder der gleiche Sachverhalt kann auf verschiedene Weise kodiert sein. Die mehrmalige Speicherung der Daten in verschiedenen Peers erzeugt Datenredundanz und in P2P-Systemen mehr Zeit- und Kommunikationsaufwand für unnötigen Datenaustausch. Solche Probleme führen dazu, dass heterogene Daten integriert werden müssen.

Datenintegration ist die Kombination von Datenbanken oder Dateien unterschiedlicher funktionaler Einheiten, die verschiedene Daten mit gleichen Merkmalen/Objekten sammeln. Dazu wurden verschiedene Lösungen vorgeschlagen. Traditionelle Ansätze dabei sind Data Warehouse (DWH) und Wrapper/Mediation (siehe Abbildung 2). Das Integrationsverfahren von DWH wird auch *In-Advance-Integration* oder *auswertungsorientierte Integration*, das Verfahren von Wrapper/Mediation wird auch *On-Demand-Integration* oder *anfrageorientierte Integration* genannt. [Vo99]

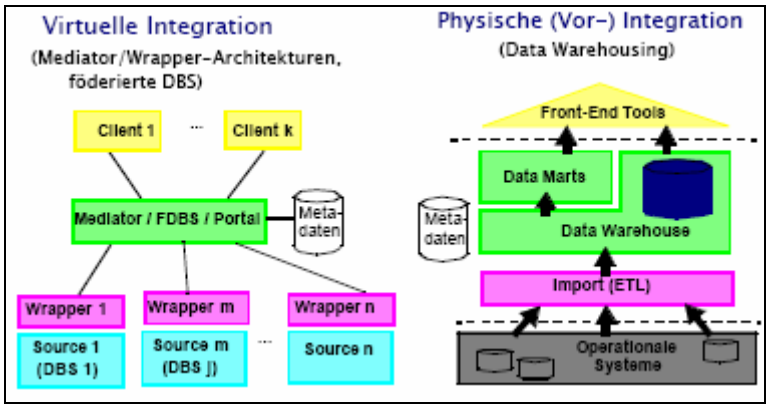

Abbildung 2: Grundlegende Alternativen zur Datenintegration [Ra02]

Beim DWH werden Daten aus mehreren heterogenen Quellen in einer zentralen Datenbank zusammengeführt (Integration) und verdichtet (Aggregation). Daten aus verschiedenen Quellen werden vereinheitlicht; dies geschieht u.a. durch eine konsistente Vergabe von Identifiern, eine einheitliche Kodierung, eine einheitliche Festlegung der Maßeinheiten von Attributen, sowie durch das Konvertieren von Datenfeldern in ein einheitliches Datenformat. Die Integration führt dazu, dass alle Daten in einer eindeutigen, allgemein akzeptierten Form

gespeichert sind. Sie erlaubt die einfache und effektive Nutzung der DWH-Daten für alle Anwender.

Die Wrapper/Mediation-Systeme haben keine materialisierte Ebene und es handelt sich bei denen um einen Typ von Software. In diesem Ansatz erfolgt die Mediation durch einen "Wrapper", ein Programm, das Daten aus den Quellen *für einzelne Clients* kombinieren (einkapseln) kann [Vo99], d.h. ein Wrapper ist immer genau einer Datenquelle zugeordnet. Eine weitere Eigenschaft des Wrappings ist, die ausschließliche Kapselung der Beziehung zwischen der Datenquelle und den Anwendungen, die den Wrapper benutzen. Somit behalten alle verwendeten Datenquellen ihre logische und physische Unabhängigkeit. Wrapper werden in einem System mit mehreren heterogenen Datenquellen eingesetzt, um jeweils eine Datenquelle in der Form einzukapseln, dass sie der gemeinsamen Schnittstelle entspricht, die das System für alle Datenquellen erwartet. I. d. R. existiert noch eine Art Vermittlungsmodul, in Form eines oder mehrerer Mediatoren, welches die Anfragen der Anwendung entgegennimmt, den Einsatz der einzelnen gewrappten Datenquellen plant, über die Wrapper auf die Datenquellen zugreift und die Ergebnisse zurückgibt [GP$^+$97]. Der Mediator wertet kodiertes Wissen über eine Menge von Daten unterschiedlicher Herkunft aus. Die resultierenden Informationen werden auf einer höheren Ebene den einzelnen Applikationen bereitgestellt.

Auf P2P-Datenverbünde kann man nicht diese traditionellen Lösungsansätze für Datenintegration übertragen und anwenden. Sie setzen ein globales Schema voraus, was bei P2P-Systemen im Allgemeinen nicht vorliegt. In P2P-Systemen ist semantische Integrität (Datenkonsistenz) nicht gesichert. Datenkonsistenz ist gegeben, wenn der Inhalt einer Datenbank oder die "Menge von Daten" alle vordefinierten Konsistenzbedingungen erfüllt. In P2P-Systemen kann nicht die "Widerspruchfreiheit" der Daten auf verschiedenen Peers garantiert werden. Dateninkonsistenz in P2P-Systemen verhindert das korrekte und zuverlässige Arbeiten mit Daten bzw. Datenbanken. Ferner ist dynamische Gestalt der P2P-Systeme ein großer Unterschied mit traditionellen Ansätzen. Im nächsten Kapitel werden Forschungsansätze zur Datenintegration in P2P-Systemen beschrieben.

3. Forschungsansätze zur Datenintegration in P2P-Systemen

Es wird intensiv auf dem Gebiet der Datenintegration in P2P-Systemen geforscht. Dabei werden die verschiedenen Probleme von P2P-Systemen und besonders der Datenintegration diskutiert. Zahlreiche Ideen sowie die Komplexität des Problems ermöglichen eine fruchtbare Untersuchung und den Vergleich zwischen verschiedenen Vorschlägen. Man kann nach unterschiedlichen Kriterien diese Forschungen klassifizieren. Einige Forschungsansätze, wie *P-Grid* [Ab$^+$01, Ab$^+$02 und Pg04] konzentrieren sich auf die Erweiterung der existierenden P2P-Systeme. Einige Forschungsansätze, wie das *semantische Web* [Be$^+$01, Be03 und FH$^+$03] und *Edutella* [BK$^+$03, Ne$^+$01] benutzen geeignete Technologien, Formate und Sprachen, die in Web festgelegt sind. Ein weiterer Ansatz, der auf ontologischen Ansätzen basiert, ist *Taxonomy* [Tz$^+$03]. Hier wird eine kurze Erläuterung über die oben genannten Ansätze geboten:

- P-Grid ist eine Selbstorganisierende Zugriffsstruktur, die eine verteilte Präfix-Baumstruktur für Routing-Suche-Anfragen verwendet. P-Grid kombiniert die Merkmale von effizienten Routing mit den Fähigkeiten der real existierenden Systeme, wie beispielsweise Gnutella und Freenet.

- *Semantische Web*: Das semantische Web möchte eine bessere Zusammenarbeit von Menschen ermöglichen. Das Web muss sich von einem persönlichen Werkzeug zu einem gemeinsamen Informationsraum verwandeln. Um dieses Ziel zu erreichen, müsste das semantische Web die Sammlung von dezentralen Webinhalten unterstützen. Das semantische Web ist nur dann möglich, wenn die Datenintegration und der Ergebnisaustausch mit anderen Programmen möglich sind. W3Consortium standardisiert in diesem Bereich die XML-basierten Sprachen, um Datenintegration und Datenaustausch zu ermöglichen.
- *Edutella*: Edutella ist ein P2P-System, das auf RDF basiert. Der hier verfolgte Ansatz löst mögliche technische Heterogenitäten und unterschiedliche Anfragesprachen durch die Festlegung geeigneter Technologien, Formate und Sprachen: Die unterliegende Technologie JXTA bestimmt die Kommunikationsplattform zwischen Peers. RDF als Beschreibungssprache für Metadaten von Web-Ressourcen beliebiger Art, wird als Format für die Beschreibung und den Austausch der Daten innerhalb der Infrastruktur benutzt. Als einheitliche Anfragesprache wird RDF-QEL (Query Exchange Language) verwendet. QEL ist prologmächtig und insbesondere von den Provider-Schemata und damit von semantischen Fragestellungen unabhängig.
- *Taxonomy:* Bei diesem Ansatz beschreiben Peers ihre Daten mit Ontologien. Taxonomy-Modell schlägt die Anwendung von einem globalen Schema und Mapping zwischen den Peers vor. Es unterstützt semantikbasierte Dienste. Der Peer benutzt seine eigenen Metadaten und Daten (eigene Taxonomy) und mit deren Hilfe eine Anfrage formuliert wird. Der andere Peer, der eine Query erhält, muss anhand seiner eigenen Taxonomy die Antwort ermitteln.

Die Ansätze, die die Datenintegration auf Instanz-Ebene vorschlagen, sind *Local Relational Model (LRM)* [Ber+01, SG+01] und *Hyperion* [Ar+03, KA+03a, KA+03b und Hy04]. Es wird in Abschnitt 3.1 und 3.3 darauf weiter eingegangen. Die Datenintegration in P2P-Systeme kann auch auf der Schema-Ebene stattfinden. Dazu werden *Both-as-View (BAV)* [MP03a, MP03b und Po03] und *Piazza* [Ha+03a, Ha+03b, Ha+03c, Ta03, MB+02, Pi04, Ha+03d und Ha+01] in Abschnitt 3.2 und 3.4 diskutiert. Hier sollen vier Forschungsansätze, nämlich LRM, Hyperion, BAV und Piazza behandelt werden. Erklärungen zu weiteren Forschungsansätzen würden den Rahmen dieser Arbeit übersteigen.

3.1 Local Relational Model (LRM)

Im LRM werden Peers in einem P2P-Netz als lokale relationale Datenbanken betrachtet, die Bekanntschaften (sog. *acquaintances*) schließen. Der Term *Bekanntschaft* wird für benachbarte Peers mit bestimmten Kriterien verwendet. Das Ziel des LRM ist es, ein formales Modell für die Einheiten einer Datenverwaltung in einem P2P-Netz zur Verfügung zu stellen. *LRM* vermeidet das Superpeermodell, welches die Ressourcen der Peers in seinem Cluster in einem Index verwaltet. Die Hauptziele des hier verwendeten Datenmodells sind, widersprüchliche Datenbanken zu berücksichtigen und die semantische Interoperabilität in der Abwesenheit eines globalen Schemas zu unterstützen.

Architektur: Datenbanken in einem P2P-System ähneln heterogen verteilten Datenbanken, oft auch Multidatenbanksysteme (MDBS) genannt. Das LRM setzt voraus, dass alle Peer-Knoten identische Schnittstelle anbieten. Dies wird LRM-Ebene benannt. Die Ebene wird auf einem lokalen Datenserver (z.B. ein DBMS) aufgesetzt. Wie in Abbildung 3 gezeigt, hat die LRM-

Ebene vier Module: User Interface (UI), Query Manager (QM), Update Manager (UM) und Wrapper.

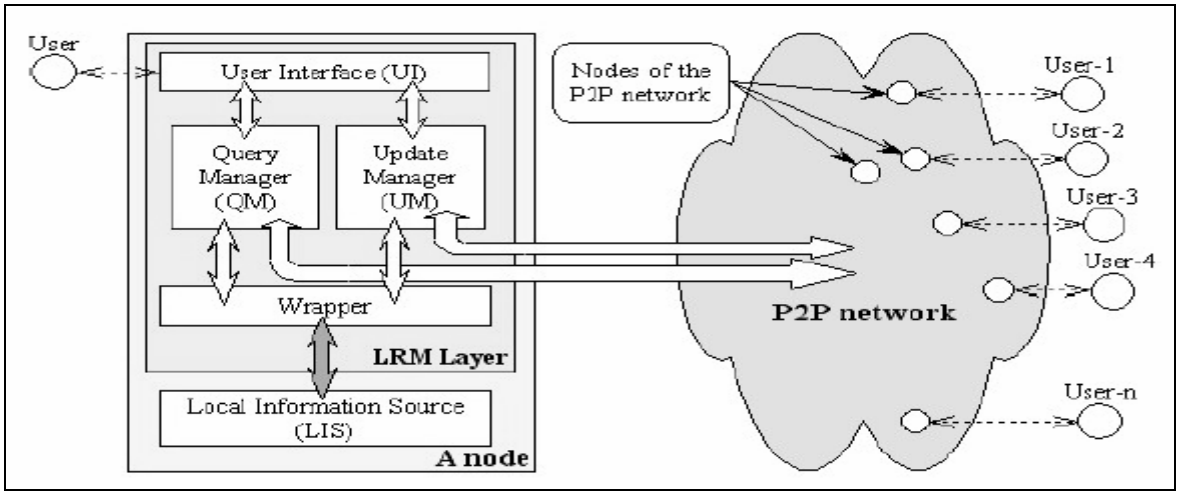

Abbildung 3. Architektur eines LRM Knotens [Ber+01]

- Das UI erlaubt einem Benutzer, Anfragen zu definieren, Ergebnisse und Meldungen von anderen Knoten zu erhalten und andere Module der P2P-Ebene zu kontrollieren.
- QM und UM sind für die Anfrage- und Aktualisierungsfortpflanzung (Update Propagation) verantwortlich. Sie verwalten Domänenbeziehungen, Koordinierungsformeln, Koordinierungsregeln, Bekanntschaften und Interessengruppen.
- Der Wrapper liefert eine Übersetzungsebene zwischen QM, UM und LIS. Peers kommunizieren durch QM und UM mit Hilfe von XML. Die Inter-Modulkommunikation basiert ebenfalls auf XML. Die Verbindung zwischen Wrapper und LIS ist anders, weil die Kommunikationssprache LIS zu der Familie der Sprachen wie SQL, http,... gehört.

Datenintegration: An Stelle eines globalen Schemas hat jeder Peer *Koordinationsformeln* und *binäre Domänenbeziehungen*.

- Koordinierungsformeln geben semantische Abhängigkeiten zwischen zwei Bekanntschaften an. Die Koordinationsformel als Hauptmechanismus der LRM-Datenintegration erklärt, wie die Daten in einem Peer sich für Daten einer Bekanntschaft spezifizieren. (Siehe Beispiel)
- Binäre Domänenbeziehungen geben an, wie Elemente einer Datenbank in Elemente einer anderen bekannten Datenbank übersetzt werden müssen. Das LRM geht von bestehenden, paarweise definierten Domainrelationen aus, die synonyme Datenelemente als Koordinationsformel beziehen und semantische Abhängigkeiten unter bekannten Datenbanken definieren.
- Durch Anfragen werden die Daten auf anderen Peers gefunden. Strategien für Anfrage- und Aktualisierungsfortpflanzung sind in einer Reihe von Koordinierungsregeln kodiert. Koordinierungsregeln beschreiben, *wann*, *wie* und *wo* eine Anfrage oder Aktualisierung verbreitet werden muss.

Die Koordinationsformel und Datensuche in LRM liegt auf der Instanz-Ebene.

Beispiel: Anhand eines Beispiels soll hier gezeigt werden, wie LRM funktioniert. Die Patienten-Datenbanken eines Hausarztes und einer Krankenhaus sollen relevante Informationen über einen bestimmten Patienten (definiert über seine Patienten-Id „Pid“) austauschen. Wenn ein neuer Patient zum Krankenhaus geht, kann die relevante Daten zwischen DB von Krankenhaus und Arzt ausgetauscht werden. Hier werden zwei Tabellen medication von Krankenhaus-DB und treatment von Arzt-DB betrachtet:

- p: Krankenhaus-DB: medication (PrescriptionID, Pid, Prod)
- d: Arzt-DB: treatment (Tid, Pid, Description, Type)
  where type ∈ {“hospital”, “home”}.

Die Krankenhaus-Datenbank identifiziert die "medication" Tabelle durch Rezept-ID (PrescriptionID) und enthält Daten über Patient-ID und Prod. Die Hausarzt-Datenbank identifiziert die "treatment" Tabelle durch Behandlung-ID (Tid) und enthält Daten über Patient-ID, Beschreibung (Description) und Typ (Type), der aus "hospital" und "home" besteht. LRM setzt für Beziehung zwischen beiden Datenbanken voraus: $(\forall i{:}x).A(x)$, d.h. für alle x in der Domäne von DB i, A(x) ist wahr.

Um "treatment" Datensatz zu erhalten, wird die Behandlung-ID (z.B. Tid=1234) benutzt. Dazu wird eine Koordinationsformel, wie:

$$(\forall p{:}y).(\forall p{:}z).(p{:}\ (\exists x).\text{medication}(x, y, z) \rightarrow d{:}\ (\exists w).\text{treatment}(w, y, z, \text{“home”})\ )$$

verwendet ("Es gibt einer Datensatz in "treatment" in der Arzt-DB für jeden Datensatz aus medication der Krankenhaus-DB.").

Um den Datensatz von Patient mit dem Identifier 1234 zu finden, wird die Pid des Patienten verwendet. Für Pid =1234 ergibt sich:

$$(\forall p{:}y).(\forall p{:}z).(p{:}\ (\exists 1234).\text{medication}(1234, y, z) \rightarrow d{:}\ (\exists 1234).\text{treatment}(1234, y, z, \text{“home”})\ )$$

Wenn die Krankenhaus-DB Daten von den Arzt-DB importiert, also die Tid, Pid und Prod, wird dies mit Hilfe einiger konkreter Elemente der Krankenhaus-DB realisiert. Daraus ergibt sich ein neues Tid für Krankenhaus-DB. Später, wenn Patient 1234 für einige Zeit im Krankenhaus behandelt wird, könnte eine andere Koordinationsformel aufgestellt werden, die nach jeder Behandlung den Datensatz aktualisiert.

Um die Erzeugung des neuen Datensatzes zu beschreiben, wird angenommen, dass das Krankenhaus keine Information über seinen neuen Patienten mit der Pid # 1234 hat und herausfinden muss, welche medizinische Vorgeschichte der Patient hat. Hier verwendet das Krankenhaus seine Bekanntschaft mit einer *Interessengruppe* von Arzt-Datenbanken, genannt allArzt-DB. allArzt-DB ist ein Peer, der jene Arzt-Bekanntschaften und eine Koordinationsformel hat, die ihr erlaubt, auf Rezeptinformation in ihren Datenbanken zuzugreifen. Zum Beispiel, wenn angenommen wird, dass allArzt-DB aus einer einzelnen Tabelle besteht:

Prescription (Tid, Name, Telefon #, DrugID, Dose, Repeats),

dann könnte die Koordinationsformel zwischen den zwei Datenbanken sein:

$\forall$Tid $\forall$Pid.(

Arzt-DB: $\exists$Tid $\exists$Pid $\exists$Description $\exists$Type.(medication (PrescriptionID, Pid, Prod) and treatment (Tid, Pid, Description, Type))

→ allArzt-DB: $\exists$name $\exists$telefon.(Prescription(Tid, Name, Telefon #, DrugID, Dose, Repeats)))

Analoge Formeln existieren für jeden anderen Bekannten-Peer von allArzt-DB. Außer als Informationsbroker zu dienen, unterstützen Interessengruppen auch Mechanismen für das

Generieren von Koordinierungsformeln aus parametrisierten gegebenen Schemainformationen für jede Arztdatenbank. Aufgrund dieser Formel, ausgewertet in Bezug auf allArzt-DB, wird eine Anfrage wie "alle Rezepte für Patienten mit Namen N und Telefon # P" in Anfragen übersetzt, die in Bezug auf Datenbanken wie allArzt-DB ausgewertet sind.

Zusammenfassung: LRM ist eine Vision, die auf einer mengenorientierten und formalen Definition basiert. Seine Architektur nutzt Ähnlichkeiten zwischen MDBS und P2P-Systeme. In einer dreischichtigen Architektur Peer, LRM-Schicht und P2P-Netz konzentriert sich LRM auf die mittlere Schicht (LRM-Schicht), die die Vermittlung zwischen lokalen Knoten und P2P-Umgebung verwirklicht. Für Datenintegration verwendet es seine eigene Terminologie. Durch Koordinationsformel findet es für jede einzelne Verbindung konkrete Lösung. Die Anfragen und Datenaustausch zwischen Peers werden durch die Koordinationsformeln ermöglicht, die Mapping zwischen Peers realisieren. Es existiert kein globales Schema, sondern es werden Mappings zwischen Datenbanken auf lokaler Ebene hergestellt.

3.2 AutoMed: Both-as-View (BAV)

BAV ist ein Teil des *AutoMed*-Projekts [Au04]. Für die Schemavermittlung in Datenintegrationssystemen werden zwei Hauptformalismen vorgeschlagen. In der ersten, global-as-view (GAV), sind die Beziehungen in dem vermittelten Schema als Menge von Sichten der Relationen in den Datenquellen definiert. In der zweiten, local-as-view (LAV), sind die Relationen in den Quellen als Sichten des vermittelten Schemas angegeben. Der LAV-Ansatz beschreibt den Inhalt von Datenquellen. Die Semantik der Formalismen ist bezüglich bestimmter Antworten auf eine Anfrage definiert. BAV ist ein Ansatz zur Datenintegration, die die vorhergehenden Ansätze von LAV und von GAV in sich vereint. Ein Vorteil von BAV gegenüber GAV und LAV ist die Möglichkeit der Entwicklung beider Schemata, einschließlich Addition oder Entfernung von lokalen Schemata.

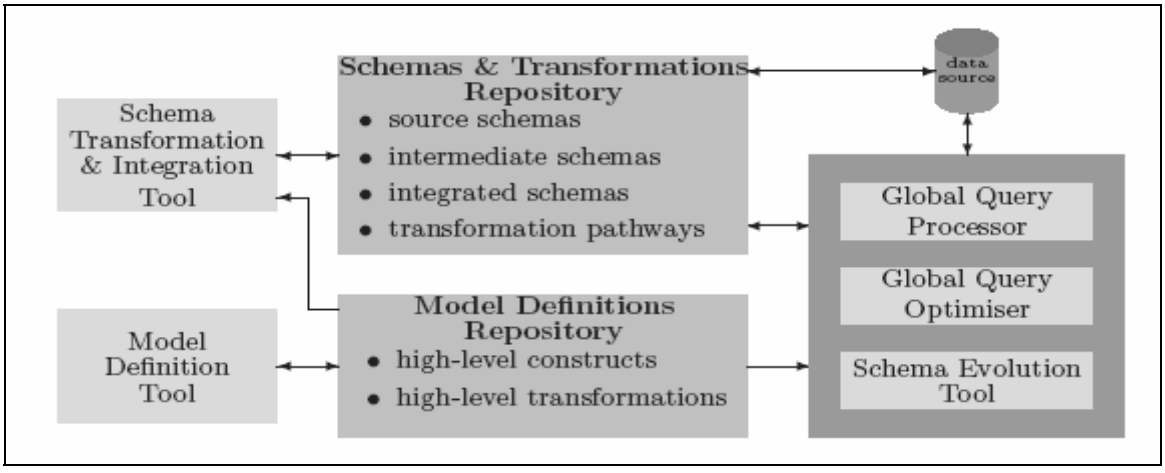

Abbildung 4. Architektur der AutoMed [MP03a]

Architektur: Abbildung 4 erläutert die Hauptkomponenten des AutoMed-Systems. Hier werden nur für den BAV-Ansatz relevante Teile behandelt. Das Modelldefinitionswerkzeug (MDT) erlaubt Modellierungskonstrukte und primitive Transformationen von höheren Modelliersprachen. Diese Definitionen sind im Model Definitions Repository (MDR) gespeichert. Das Schemas & Transformations Repository (STR) speichert die Schemata und die Transformation zwischen ihnen. Das Schema Transformation & Integration Tool erlaubt die Schaffung von neuen Zwischen- oder globalen Schemata zusammen mit den entsprechenden Transformation Pathways. Der globale Query Processor übernimmt die

Ausführung von globalen Anfragen mit Hilfe der Schemata und des Transformation Pathways beim STR. AutoMed übernimmt gegenwärtig einen GAV-Ansatz dazu, dass Anfragen verarbeitet, aber ein LAV-Ansatz wäre auch möglich. STR ist mit Hilfe BAV-Ansatz realisierbar.

Datenintegration: BAV erweitert eine Datenmodellierung, um eine bessere Grundlage für Datenintegration zu schaffen. Im BAV wurden die Modellierkonstrukte der höheren Ebene der Datenmodelle in HDM (Hypergraph Data Modell)-Terms dargestellt (z.B. relational, Objekt-Orientiert, semistrukturiert, UML, XML, RDF). HDM liefert ein gemeinsames Datenmodell, eine einheitliche Semantik für die Modellierkonstrukte. HDM vermeidet die semantischen Abweichungen, die zwischen Konstrukten der Modellierungssprachen höherer Ebenen auftreten können. Dieser Ansatz erlaubt Konstrukten von verschiedenen Modellierungssprachen, gemischt innerhalb desselben intermediaten Schemas während der Schematransformation/des Integrationsprozesses zu sein.

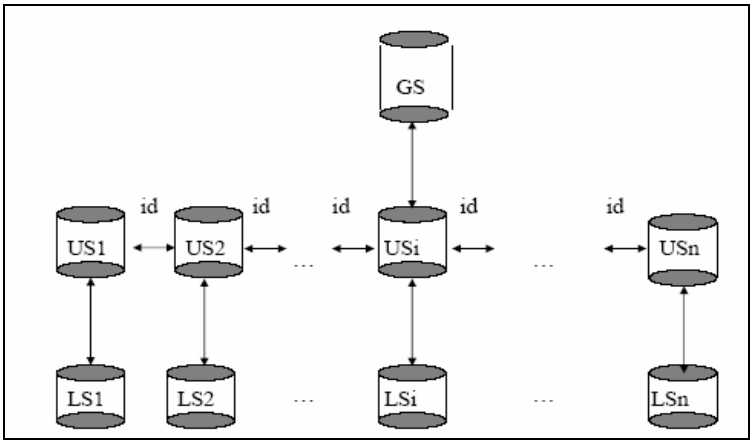

Abbildung 5. Schema Transformation/Integration Netzwerk in AutoMed [Po03]

- *GS ist ein globales Schema*
- *LS_1, …, LS_n sind lokale Schemata*
- *US_1, …, US_n sind union-compatible Schemata*
- *die Transformation Pathways zwischen jedem Paar LS_i und US_i können aus add, delete, rename, expand und contract bestehen*
- *die Transformation Pathways zwischen US_i und GS sind ähnlich*
- *die Transformation Pathways zwischen jedem Paar union-compatible Schemata besteht aus id Transformationsschritten*
- *Schema Transformation Pathways erfasst mindestens die verfügbare Information von GAV oder LAV*

Die BAV-Methode für die Datenintegration basiert auf dem Gebrauch von bidirektionalen Schematransformationen. In BAV wird eine Folge bidirektionaler Schematransformationen abgebildet, die als *Pathway Transformation* bezeichnet wird. BAV nutzt das *Superpeer Schema* für sein globales Schema.

Command	Beschreibung	Beispiel
contractTable	setzt zwei Tabellen in eine Ober- bzw. Untermenge Beziehung	contractTable (<<patient, ni, name, sex, gp>>, Void, <<allPatients, ni, name, sex, gp>>)
extendTable	Um eine neue Tabelle aus einer bestehenden abzuleiten	extendTable (<<allPatients, ni, name, sex, gp>>, <<patient, ni, name, sex, gp>>, Any)
extendAtt	Um ein neues Attribut aus einem bestehenden abzuleiten	extendAtt (<<patient, hid>>, Void, Any)

Tabelle 1. Ausgewählte Transformationen von BAV

BAV hat eine Folge von primitiven Transformationen, die aus add, delete, rename, expand und contract bestehen. In Tabelle 1 stehen ein paar ausgewählte Transformationen, die auf der Basis dieser primitiven Transformationen entwickelt wurden. contractTable und extendTable sind Transformationen, die auf Rel-Konstrukt und seine verwendete Att-Konstrukte aufgebaut sind. Eine ähnliche Vorgehensweise gilt auch bei den Attributen.

Von diesem Pathway ist es möglich, eine Definition des globalen Schemas als Sicht aus dem lokalen Schema zu extrahieren, und es ist auch, möglich Definitionen des lokalen Schemas als Sicht aus dem globalen Schema zu extrahieren.

Beispiel: Anhand eines Beispiels soll hier gezeigt werden, wie BAV funktioniert [MP03b]. Peer-Schemata PS_1, PS_2 werden betrachtet, mit:

PS_1 patient (hid, ni, name, sex, age, gp)

PS_2 patient (ni, fName, lName, sex, address)

Peer-Schema PS_1 ist ein Schema für die Patientendatenbank eines Krankenhauses. Jedem Patienten ist ein eindeutiger hospital identifier (hid) zugewiesen, und die anderen Einträge beschreiben durch Sozialversicherungsnummer (nationale insurance - ni), name, sex, age, Allgemeinarzt (General Practitioner - gp) den Patienten. Das Peer-Schema PS_2 identifiziert die Patienten nach ihrer ni Nummer und beschreibt sie durch first Name (fName), Nachname (lName), sex und address. Ein mögliches Super-Peer-Schema wird in SPS_1 gegeben:

SPS_1 allPatients (ni, name, sex, gp)

Es wird angenommen, dass das Krankenhaus, das PS_1 besitzt, die Information in seiner Tabelle mit anderen Peers austauschen möchte, welches als PS1.patient bezeichnet wird. Jeder Patientendatensatz in PS_1 könnte an einen anderen Peer gesendet werden, der SPS_1 entspricht. Umgekehrt könnte einer Patientendatensatz von einem anderen Peer, der SPS_1 entspricht, in PS_1 importiert werden. Der BAV-Pathway von PS_1 zu SPS_1 ist wie folgt:

① contractAtt (<<patient, age>>, Void, Any)

② contractAtt (<<patient, hid>>, Void, Any)

③ extendTable (<<allPatients, ni, name, sex, gp>>, <<patient, ni, name, sex, gp>>, Any)

④ contractTable (<<patient, ni, name, sex, gp>>, Void, <<allPatients, ni, name, sex, gp>>)

Die ersten zwei Schritte entfernen Informationen über Alter und Krankenhaus-ID aus dem Schema. Der nächste Schritt zeigt auf, dass SPS1.allPatients eine Obermenge von PS1.Patient ist, während der letzte Schritt zeigt, dass PS1.patient eine Teilmenge von SPS1.allPatients ist.

Um genauer die Anwendung der Pathways darzustellen, wird angenommen, dass der Datensatz des Patienten Joe Bloggs in PS_1 existiert. Nun stellt sich die Frage, wie der Datensatz von Joe Bloggs zu PS_2 geschickt werden soll. Dazu braucht man eine Aktualisierungsanfrage, die die Pathways ① bis ④ anwenden. Z.B. update-request

instert patient (10000, ´ZS341234P´, ´Joe Bloggs´, ´M´, 56, ´Davies´)

ist von PS_1 zu PS_2 gesendet. Transformationen ① und ② konvertieren die Records zu:

patient (´ZS341234P´, ´Joe Bloggs´, ´M´, ´Davies´)

Es erzeugt eine mit einander vereinbare Vereinigung mit allPatients in Schema SPS_1. Die Transformation 3 erklärt, dass diese patient untere Schranke der allPatients in das Superpeer SPS_1 ist. patient benutzt das Superpeer Interpretation, gibt die Range der Werte als:

[allPatients (´ZS341234P´, ´Joe Bloggs´, ´M´, ´Davies´),

allPatients (´ZS341234P´, ´Joe Bloggs´, ´M´, ´Davies´)]

Zusammenfassung: BAV ist ein Datenintegrationsansatz, der in sich die LAV- und GAV-Ansatz vereint. BAV ist ein wichtiger Teil des AutoMed-Projekts, das für Schema-Mediation entwickelt wurde. Durch das Modellierungswerkzeug sind Modellierungskonstrukte und primitive Transformationen möglich. Ferner werden neue Schema mit Hilfe vorgegebener Schemata von Peers konstruiert. Die Transformation zwischen lokalen Schemata und globalem Schema wird durch Transforamtionspathways realisiert und dadurch werden Anfragen von verschiedenen Peers ermöglicht. Für die Aktualisierung, Stellen einer Anfrage und SchemaTransformation bieten Pathways eine flexible Möglichkeit.

3.3 Hyperion

Die Hauptziele des Projekts sind: die Definition einer P2P-Datenverwaltungsarchitektur; die Untersuchung überlebensfähige Datenintegration, Datenaustausch und Mappingsmechanismen; die Entwicklung von Algorithmen für die effiziente Suche, die effiziente Retrieval und den effizienten Austausch von Daten unter Peers. [Hy04]

Architektur: Hyperion definiert für eine Architektur, die Koordinierung zwischen Peers erlaubt, wie ihre Daten entstehen. Abbildung 6 beschreibt die Architektur. Diese Architektur verwirklicht die logische P2P-Datenkoordinierung. Hyperion ist ein PDBMS (Peer Datenbank Management System). Ein PDBMS besteht aus drei Hauptkomponenten: eine Schnittstelle (P2P-API), eine P2P-Ebene und ein DBMS. Die letzte Komponente enthält die lokalen Daten und ein Mapping der Mengen, Tabellen und Ausdrücke, die verwendet werden, um Daten mit einem anderen Peer zu tauschen, während das P2P-API die Benutzerschnittstelle zum ganzen System ist. Durch das P2P-API ist der Benutzer unter anderem fähig, Anfragen zu stellen und anzugeben, ob diese nur am Ort ausgeführt werden sollen oder sie auch die Daten mit Residenz in anderen Peers betrachten sollten. Die P2P-Ebene in Abbildung 6 ist die Schlüsselkomponente des ganzen Systems. Diese Ebene implementiert die verlangte Funktionalität, dass Peers Daten gemeinsam benutzen und koordinieren, ohne ihre eigene Autonomie zu gefährden. Ihre Grundmodule und ihre Rollen sind wie folgt:

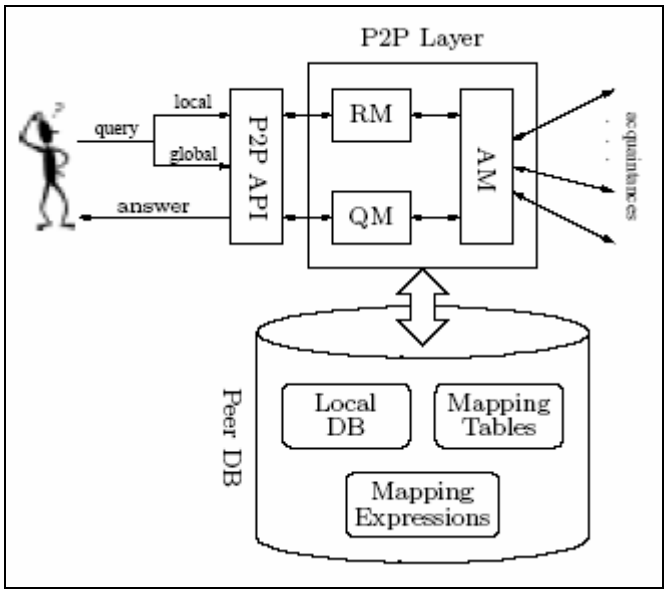

Abbildung 6. Architektur für ein PDBMS [Ar+03]

- *Acquaintance Manager (AM):* Dieses Modul ist für die semiautomatische Einführung der Bekanntschaft unter PDBMS verantwortlich. Eine Bekanntschaft wird zwischen zwei PDBMS's eingeführt, wenn eine Menge von Mapping-Tabellen oder -Ausdrücken zwischen ihnen hergestellt wird. Die Erstellung dieser Konstrukte hängt bei AM sowohl von der Benutzereingabe als auch von automatisierten Werkzeugen für die Schaffung von Mapping-Tabellen ab. Allerdings ändern sich Mapping-Tabellen und Ausdrücke wie in jeder Bekanntschaft und tragen zur dynamischen Natur des Systems bei.
- *Query Manager (QM):* Eine wichtige Annahme im Hyperion-Ansatz ist, dass jede Anfrage in Bezug auf das Schema von einem einzelnen Peer definiert ist. Eine Hyperion-These ist, dass ein Benutzer nur bzgl. der lokalen Darstellung sachkundig sein muss. In Bezug auf die Ausführung jedoch sind die Anfragen in zwei Kategorien eingeteilt. Eine lokale Anfrage wird mit Hilfe der Daten beim lokalen Peer ausgeführt, während eine globale Anfrage das P2P-Netz verwendet, um am Ort wieder aufgefundene Daten mit weiteren Daten zu ergänzen oder zu vereinen, die sich in anderen Peers befinden. Im Falle von globalen Anfragen formuliert der QM eine lokale Anfragendarstellung neu. Um dies zu tun, verwendet der QM die Anwendungen AM.
- *Rule Manager (RM):* Das Ziel des RMs ist, Konsistenzmethoden zwischen Peers durchzusetzen. Die Methoden werden deklarativ durch das P2P-API angegeben.

Datenintegration: Hyperion stellt kein Schema, sondern berücksichtigt das Problem als Abbildung der Daten in P2P-Systemen. Solche Systeme hängen von einfachem Wertsuchen ab, damit bestimmte Daten ausfindig gemacht werden können. Jedoch kann es sein, dass verschiedene Peers verschiedene Werte verwenden, um dieselben Daten zu identifizieren oder zu beschreiben. Um diese unterzubringen, hängen P2P-Systeme oft von Mapping-Tabellen ab, die Paare entsprechender Werte für Suchdomänen auflisten, die bei verschiedenen Peers verwendet werden.

Eine Mapping-Tabelle stellt Fachwissen über eine Menge von verwandten Domänen dar und ist normalerweise manuell von Betreuern konstruiert. Die Mapping-Tabellen können verwendet werden, um Werte nicht nur innerhalb einer einzelnen Domäne, sondern auch horizontal unter verschiedenen Domänen zu verbinden. Eine Mapping-Tabelle besteht im Allgemeinen aus zwei getrennten Mengen von Attributen X und Y. Ein Tupel (x, y) in der Mapping-Tabelle zeigt an, dass der Wert x mit y verbunden wird. Auf diese Art gibt eine Mapping-Tabelle gültige Zuordnungen zwischen Werten in zwei Peers an.

Ferner sind Mapping-Tabellen ein angemessenes Werkzeug, das bei P2P-Systemen zu verwenden ist, da sie die Autonomie der Peers respektieren. Hyperion schlägt totale Abwesenheit jeder zentralen Berechtigung oder Steuerung vor. Es gibt kein globales Schema. Um korrekt Daten umzustrukturieren und abzubilden, müssen die Quellen bereit sein, wenigstens Teile ihrer Schemata gemeinsam zu benutzen und beim Einführen und dem Verwalten der Übersetzungsprogramme und Anfragen zusammenarbeiten.

Beispiel: Es wird wieder das Anfangsbeispiel (Abschnitt 1.1) der Gen-DB (GDB), der SwissProt-DB und der MIM-DB betrachtet. Um den notwendigen Grad von Integration zu erreichen, verwendet Hyperion wieder *Mapping-Tabellen*. Die Mapping-Tabellen können weltweit von biologischen Datenquellen verwendet werden, um Gen-Daten in einer Quelle z.B. auf die verwandten Proteindaten in einer anderen Quelle zu beziehen. Eine Mapping-Tabelle ist nicht unbedingt eine Funktion. Es kann viele Proteine geben, die mit einem Gen verbunden sind. Sogar eine Mapping-Tabelle, die Gen-Identifier beinhaltet, kann eine

mehrdeutige Korrespondenz sein. Dies tritt oft in biologischen Quellen auf, wo es Aliasnamen für denselben Identifier geben kann. Da Identifier aktualisiert werden, kann es sein, dass alte Identifier behalten werden müssen, wenn sie sich zum Beispiel auf den Inhalt von statischen Quellen beziehen, die veraltete Namen und Identifier enthalten.

Ein Beispiel für solch einen Tabelle wird in Abbildung 7 gezeigt. Die erste Tabelle (Peer 1) zeigt, dass Paare von Genen-ID und Proteinen-ID zusammen durch eine genetische Störung (MIM-ID) in Relation gesetzt werden können. Tabelle von Peer 2 verbindet durch GDB_id und SwissPort_id Gene mit Proteinen, während die Tabelle von Peer 3 direkt Gene mit genetischer Störung assoziiert. Jede Tabelle kann von verschiedenen Betreuern mit unterschiedlichen (möglicherweise überlappenden) Kenntnissen von den zugrunde liegenden Domänen gebaut worden sein. Mapping-Tabellen respektieren die Autonomie der Quellen, die sie verbinden. Die Mapping-Tabelle von Peer 2 kodiert nur die Tatsache, dass ein Domänenexperte bestimmt hat, dass bestimmte Gene mit bestimmten Proteinen verwandt sind. Solche Informationen sind notwendig, um wirksam Suchen über Peers auszuführen.

Peer 1: GSM-ID

GDB_id	SwissProt_id	MIM_id
GDB:120231	P21359	162200
GDB:120231	O00662	193520
GDB:120232	P35240	101000

Peer 2: GS-ID

GDB_id	SwissProt_id
GDB:120231	O00662

Peer 3: GM-ID

GDB_id	MIM_id
GDB:120233	162030

Abbildung 7. eine Anfangsmenge von Mapping-Tabellen mit Identifier 3 Datenbanken [KA+03b]

Benutzer kann Tabelle von Peer 3 direkt in einer Anfrage zum Finden aller genetische Störungen mit einer spezifizierten Gen verwendet. Hier wird Tabelle von Peer 1 GSM-ID, Tabelle Peer 2 GS-ID und Tabelle Peer 3 GM-ID genannt. Eine einfache Anfrage wird darunter gezeigt:

```
Q1: select  MIM_id                  oder   Q2: select  SwissPort_id
    from    GM-ID                          from    GSM-ID
    where   GDB_id = 120233                where   GDB_id = 120231  and MIM_id = 193520
```

Zusammenfassung: Hyperion definiert einer P2P-Datenverwaltungsarchitektur, die Datenintegration, Datenaustausch und effiziente Suche von Daten verwirklicht. Hyperion hat wie LRM eine 3-Schichtige Architektur, die aus P2P-API, P2P-Layer und Peer-DB besteht. P2P-Schicht als Kern dieser Architektur übernimmt die Erstellung der Mapping-Tabellen, und der Bekanntschaften sowie die Aktualisierung derer. Hyperion unterscheidet zuerst zwischen lokalen und globalen Anfragen und schlägt für jede einzelne Anfrage eine konkrete Lösung vor. Diese reduziert die Last des P2P-Netzes. Die Datenintegration wird durch die Mapping-Tabellen verwirklicht, die auf der Instanz-Ebene hergestellt werden. Für die Realisierung der Mapping-Tabellen verwendet Hyperion ein semiautomatisches Verfahren. Insgesamt steht für Hyperion-Ansatz die Autonomie des Peers im Mittelpunkt.

3.4 Piazza

Das Piazza-Projekt ergab sich als eine Weiterentwicklung des Sagres-Systems [Sa04]. Ziel des Piazza-Projekts ist der Aufbau einer dezentralen Architektur für Nutzung und Verwaltung der Daten in verteilter und erweiterbarer Art (Peer-Datenmanagementsystem (PDMS)). In dieser Architektur kann jeder Benutzer neue Daten, Schemainformationen oder sogar Abbildungen zwischen verschiedenen Schemata beitragen.

Architektur: PDMS unterstützt jedes beliebige Netz von Beziehungen zwischen Peers, aber die Neuheit von PDMS liegt in der Fähigkeit, transitive Beziehungen zwischen den Schemata des Peers auszunutzen. Der Aspekt des PDMS soll "semantische Mediation" zwischen einer Umgebung (Environment) von Tausenden Peers, jedes mit seinen eigenen Schemata, zur Verfügung stellen.

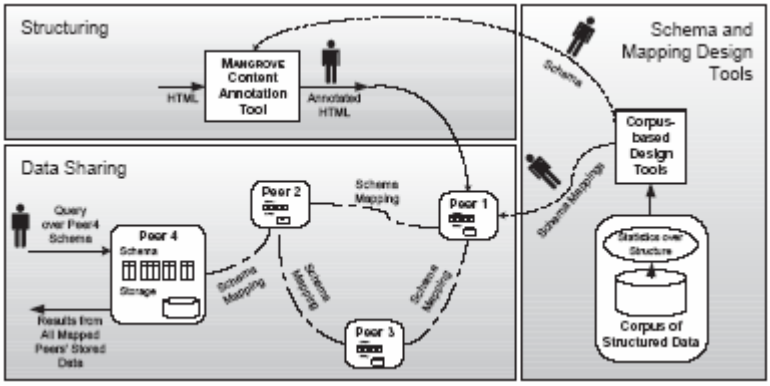

Abbildung 8. Das REVERE-System [Ha+03e]

Abbildung 8 zeigt das Piazza-System in seiner Umgebung, die unter dem REVERE-System [Ha+03e] vorgestellt wurde. Das REVERE-System existiert für die Beschreibung oder das Strukturieren von vorhandenen Daten, P2P-Daten, die eine Umgebung gemeinsam benutzen. Das Piazza-System (Abbildung 8 - unten links) ist für das PDMS verantwortlich. Die Benutzer können Anfragen in einigen der Schemata von Peers stellen, und Antworten von allen Peers zu erhalten. Die Werkzeuge helfen Benutzern und beraten Benutzer für das Definieren von Schemata und Mappings, die ein Korpus von strukturierten Daten benutzen.

Piazza ermöglicht die Entwicklung, die Abbildung und die Verwaltung der Daten in einer dezentralen und Ad-hoc-Art. In einem PDMS können Peers als Datenprovider, logische Vermittler oder bloße Anfrageknoten dienen. Semantische Mappings zwischen ungleichartige Schemata werden direkt zwischen zwei (oder einer kleinen Menge von) Peers gegeben. Mit Hilfe dieser transitiv semantischen Mappings können Peers an beliebigen Stellen im System relevante Daten benutzen. Folglich können Anfragen in einem PDMS mit Hilfe des lokalen Schemas vom Peer und ohne Kenntnis der Schemata anderer Peers gestellt werden. Mit anderen Worten heißt das: Anfragen werden über die Relationen von einem bestimmten Peer-Schema aus gestellt.

Datenintegration: Piazza nimmt an, dass die Teilnehmer an gemeinsamer Nutzung von Daten interessiert und bereit sind, *paarweise* Mappings zwischen ihren Schemata zu definieren. Ein Piazza-System besteht aus einem Netz von verschiedenen Peers, wo jede Ressource zum allgemeinen System beiträgt. Die von einem Peer beigetragenen Ressourcen schließen eins oder beide von den Folgenden ein:

(1) Daten oder *extensionale* Daten, z.B. XML- oder RDF-Dateninstanze,

(2) Metadaten, z.B. XML Schema oder OWL Ontologien.

Außerdem können Knoten Daten aus zwischengespeicherten Antworten anderen Knoten zur Verfügung stellen. Wenn ein neuer Peer (mit Dateninstanz oder Schema) dem System hinzugefügt wird, wird er semantisch auf irgendeinen Teil des vorhandenen Netzes bezogen. Eine Anfrage in Piazza wird immer aus der Perspektive von dem Schema eines gegebenen Peers formuliert, welches die bevorzugte Terminologie des Benutzers definiert. Wenn eine

Anfrage gestellt wird, liefert Piazza Antworten, die alle semantisch in Beziehung stehenden XML-Daten innerhalb des Systems verwendet.

Um Daten aus anderen Peers auszunutzen, muss semantische Mappings zwischen den Peers erzeugt werden. Mappings in Piazza sind spezifiziert zwischen einer kleinen Anzahl von Peers, normalerweise Paaren. Auf diese Weise ist Piazza in der Lage, zwei ziemlich verschiedene Methoden für die zuvor erwähnte semantische Vermittlung zu unterstützen:

- *vermittelndes Mapping*, wo Datenquellen durch ein vermittelndes Schema oder Ontologie verbunden sind,
- und *Punkt-zu-Punkt-Mapping*, wo Daten danach beschrieben werden, wie sie übersetzt werden können, um dem Schema eines anderen Peers zu entsprechen.

Der tatsächliche Formalismus für die angegebenen Mappings hängt von den Arten von Peers ab, die Piazza abbildet. Es gibt drei Hauptfälle, je nachdem ob wir zwischen Paaren von OWL/RDF-Knoten, zwischen Paaren von XML/XML-Schema-Knoten oder zwischen Knoten von verschiedenen Typen abbilden. Eine Beschreibung darüber findet man in [Ha$^+$03d]. Piazza kombiniert und verallgemeinert die GAV- und LAV-Datenintegrationsformalismen. Sie dehnt sie auf die XML Welt auf eine Weise aus, die die Auswertung in einem verwendbaren Rahmen hält. In einheitlicher Art verbindet der Piazza-Algorithmus GAV- und LAV-Ansätze (sog. GLAV-Ansatz). [Ha$^+$03d]

Beispiel: Wie in Abbildung 9 gezeigt, definieren einige Universitäten Mappings unter ihren Schemata, so dass sie transitiv zusammenhängend sind. Jetzt, da andere Universitäten in diese

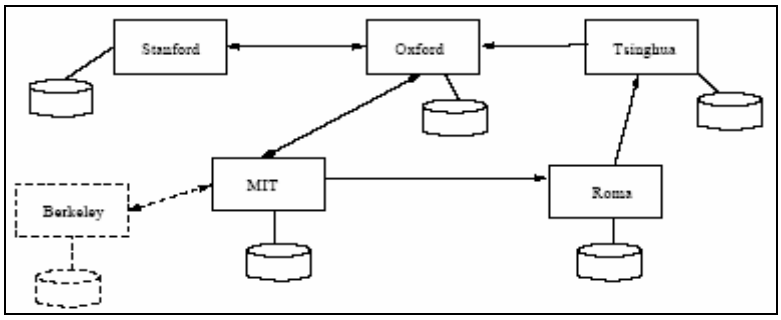

Abbildung 9: PDMS für Universitätsbeispiel. Die Pfeile entsprechen Schema-Mappings zwischen Peers. Kein zentrales vermittelndes Schema ist notwendig. Solange der Mapping-Graph zusammenhängend ist, kann jeder Peer auf Daten an jedem anderen Peer durch Folgen von Schema-Mappings, sog. „links", zugreifen.

Koalition eintreten möchten, formen sie Mappings zu dem System, das ihrem eigenen am ähnlichsten ist (z.B. Trento Map zu Rom). Sie werden transitiv an die anderen angeschlossen.

Das zentrale Problem für PDMS ist das der Anfragebeantwortung (Query Answering): jede Benutzeranfrage ist über ein logisches Peer-Schema gestellt und muss neu geschrieben werden, so dass es sich letztlich jeweils auf gespeicherte Relationen auf den verschiedenen Peers bezieht. Für die Datenintegration nutzt Piazza die GLAV-Architektur. [Ha$^+$01]

Piazza entwickelt gegenwärtig eine Mappingsprache, damit aufeinander bezogene XML-Daten und eine Menge von Neuformulierungsalgorithmen sich ihrer bedienen können. Diese Sprache beginnt mit einer "*Maske*" (template), einem definierten Schema des Peers. Der Datenbankadministrator des Peers kommentiert Teile dieser Maske mit Anfrageinformationen, um zu definieren, wie man die erforderlichen Daten aus Quellenbeziehungen oder anderen Peer-Schemata extrahieren kann. Ein Beispiel von Peer-Schemata (in XML-DTD-Form) sieht man für Berkeley und MIT in Abbildung 10 [Ha$^+$03d].

Berkeley peer schema (XML DTD):
Element schedule(college*)
Element college(name, dept*)
Element dept(name, course*)
Element course(title, size)
MIT peer schema:
Element catalogue(course*)
Element course(name, subject*)
Element subject(title, enrollment)

Abbildung 10: Peer-Schemata von Berkeley und MIT

Der Piazza-Ansatz zeigt Ähnlichkeiten mit der XDR-Mapping-Darstellung von Microsoft SQL Server und der von IBM verwendeten XML-Extender Annotation, aber Piazza verwendet tatsächlich eine Teilmenge von XQuery, um die Angleichungen von XML-Daten zu einer XML-Schema und nicht von relationalen Daten an eine XML-Schema zu definieren. Eine Lösung für das Mapping zwischen dem Berkeley-Peer Schema und dem MIT-Schema sieht man hier (Abbildung 11):

```
<catalog>
   <course> {$c = document("Berkeley.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>
```

Abbildung 11: Mapping zwischen dem Berkeley-Peer Schema und dem MIT-Schema

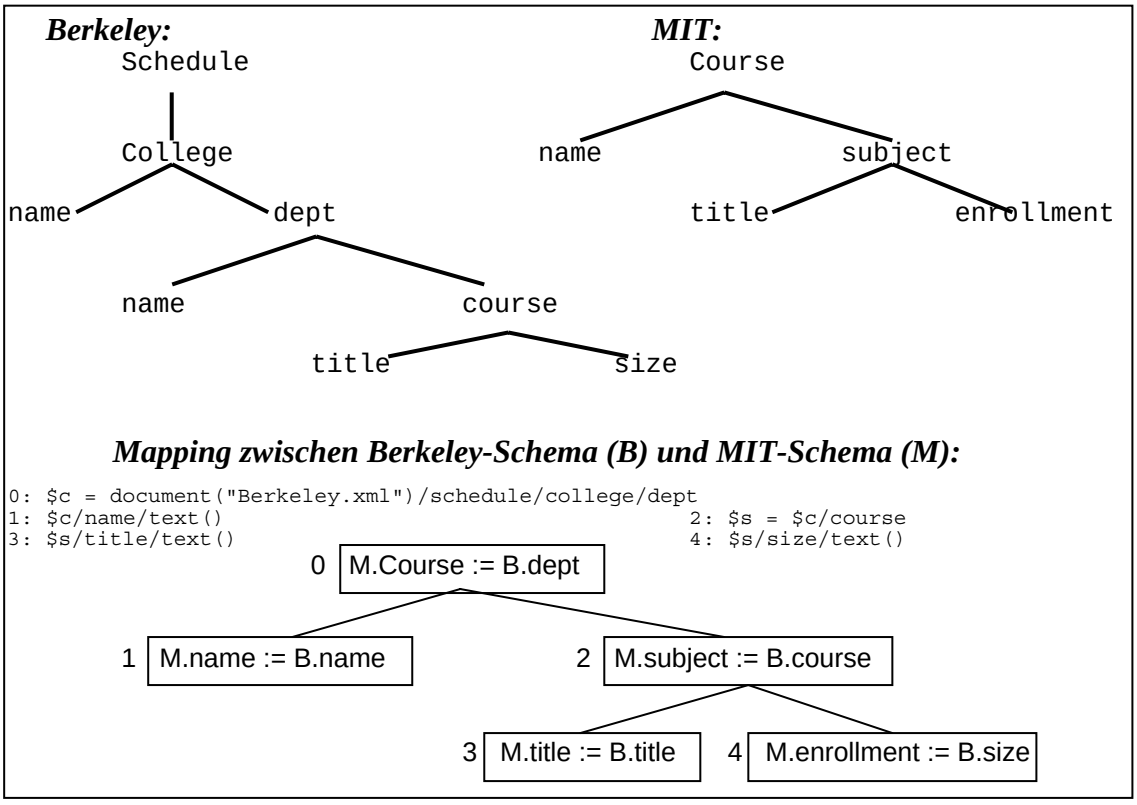

Abbildung 12: Visualisierte Mapping zwischen dem Berkeley-Peer Schema und dem MIT-Schema

In diesem Mapping ist zu sehen, dass die Maske zum MIT-Schema passt. Die durch geschweifte Klammern (neben Elemente: `<course>` und `<subject>`) abgegrenzten Anmerkungen beschreiben in Anfrageform, wie Variable, denen Dollarzeichen vorangestellt sind, an Werte im Ausgangsdokument gebunden sind; jede Bindung führt zu einer Spezialisierung vom jeweiligen Teil der Schablone mit der Anmerkung. Abbildung 12 visualisiert dieses Mapping.
Piazza hat eine vorläufige Version (und Implementierung) der Mapping-Sprache, die eine hierarchische XML-Konstruktion und beschränkte Pfadausdrücke unterstützt, vermeidet aber die meist komplexen Merkmale von XQuery (eine spezielle Erweiterung stellt Piazza als PPL-Sprache vor). Für das oben genannte Beispiel kann man folgendes XQuery benutzen (Abbildung 13):

```
<result> {
   for   $title   in schedule/college/dept/course/title/text(),
         $size    in schedule/college/dept/course/size/text(),
   where $title = "P2P- Data Management"
   return
   <size> { $size } </size>
</result>
```

Abbildung 13: Eine Anfrage

Die Anfrage sucht nach Sub-Elemente "title" und "size" und wenn der Titel "P2P-Data Management" ist, gibt sie die Einschreibungskapazität der Kurs (bzw. Kurse) zurück.

Zusammenfassung: Das Ziel von Piazza ist der Entwurf einer skalierbaren Datenintegration, die auf dezentrales Schema Mediation und Mapping zwischen Schemata basiert. Diese Mappings werden paarweise definiert und nutzen Metadaten und semantische Beziehungen zwischen Schemata. Diese Architektur bietet eine potenzielle Möglichkeit für die Erweiterung dieses Projekts, um im Bereich des semantischen Webs Grundanforderung zu erfüllen. Piazza unterstützt dies durch die Implementierung des Projekts mit XML, XML-Schema und XQuery.

4. Vergleich zwischen Forschungsansätzen

Mit Hilfe von Tabelle 2 kann man sehen, dass die Lösungsansätze für P2P-Datenintegration unter Hauptaspekten wie zentralisierte/dezentralisierte Architektur, Integration auf Schema-/Instanzebene und Entscheidung über Query- und Programmiersprache usw. klassifizierbar sind. Für fast alle Ansätze spielen Markup-Sprachen, wie XML und die Querysprache SQL eine große Rolle. Manche Projekte wie Hyperion, BAV und Piazza erweitern auf dieser Basis ihre speziellen Sprachen. Es kann eine Tendenz in diesem Bereich beobachtet werden. Die Verwendung von Markup-Sprachen und SQL-"Dialekten" hat gute Chancen in der Zukunft eine entscheidende Rolle bei der Datenintegration zu spielen.

Darüber hinaus haben Forschungsprojekte voneinander gelernt. Die LRM hat viel zu Hyperion und BAV beigetragen. BAV nutzt die Vorteile von Piazza und LRM durch die Verwendung eines gemeinsamen virtuellen Superpeer-Schemas, hat aber keinen physischen Superpeer-Knoten. Bei Hyperion wurden die verwendeten Schlüsselelemente für das Angeben und das Verwalten von logischen Metadaten beschrieben, die Datenbenutzen und Koordinierung zwischen PDBMS ermöglichen. Kurz gesagt versuchen die neuen Lösungsansätze Vorteile von vorherigen Lösungen zu kombinieren und ihrer Nachteile zu vermeiden. Ein gutes Beispiel dafür ist BAV.

	LRM	Hyperion	BAV	Piazza
Vorschlag	[Ber+01] und [SG+01]	[Ar+03], [KA+03a], [KA+03b] und [Hy04]	[MP03a], [MP03b] und [Po03]	[Ha+03a], [Ha+03b], [Ha+03c], [Ta03], [MB+02], [Pi04], [Ha+03d] und [Ha+01]
Architektur	Dezentralisiert, Ähnlich: MDBS, vermeidet Superpeer	Dezentralisiert, leichte Koordinierung zwischen Peers, lokale Anfragen werden nur mit Hilfe der Daten des lokalen Peers ausgeführt, während eine globale Anfrage das P2P-Netz verwendet	Hybrid, Nutzung von Superpeer-Schemata für globales Schema	Dezentrale Schema-Mediation
Voraussetzung der Architektur	Alle Peers haben eine identische Struktur, die aus einer LRM-Ebene besteht	Benutzer benötigt nur von lokalem Schema zu wissen.	Schema	Schema
Begriff/ Terme/ Eigenschaften	Acquiantance Superpeer (nicht verw.) Koordinierungsformel Querysemantik Binäre Domänenbeziehung	Mapping Table Mapping Expression P2P-Services (P2P-Querying, P2P-Coordination) GLAV	GAV LAV Pathway	PDMS Schema Mediation Peer Schema Peer Relation GLAV
Auf welcher Ebene?	Kein globales Schema, sondern Instanzebene	Instanzebene, kein globales Schema, keine vorübergehende Beteiligung von PDB	Schema Mediation	Schema Mapping
Datenmodell/ Erweiterung von	Rel. DB	Erweiterung DBMS → PDBMS	HDM-Term	XML
Programm/ Query Language	XML, SQL	SQL, ECA	HDM, XML, RDF, IQL (intermediate query Language), UDDI	XML
Datenintegration	Eine Logik für P2P-Datenintegration Peers sind autonom und berücksichtigen nur ihren direkten Bekanntschaften	einfaches Wertsuchen durch Mapping der Daten Peers nutzten Mapping-Tabelle	Umschaltbare Schema-Transformation	Paarweises Mapping zw. Schemata Nutzung von Metadaten Komb. von GAV & LAV ⇒ GLAV

Tabelle 2. Vergleich zwischen fünf Forschungsansätze

5. Zusammenfassung

Die Datenintegration beschränkt die Autonomie des Peers und definiert eine domainspezifizierte P2P-Umgebung, die das Idealbild der P2P-Systeme verletzt. Zwischen den existierenden P2P-Systemen und den Erwartungen der Forschung besteht noch eine große Kluft. Systeme wie Napster, Gnutella, Freenet usw. bieten für Datenmanagement und Datenintegration kaum Möglichkeiten. Für die Datenintegration setzt man folgende Elemente voraus:

a. inhaltlich gleiche oder vergleichbare Daten. Dies führt dazu, dass semantische Ansätze und passende Werkzeuge für die Erzeugung von Metadaten entwickelt werden. Ein guter Startpunkt können die XML/RDF-Sprachen und die Entwicklung der Ontologie-Sprachen wie OWL bzw. OIL/DAML sein.

b. gemeinsame oder interoperable Sprachen sind für Datenmodellierung bzw. Anfragesprachen notwendig, z.B. die Kombination zwischen JAVA, XML und SQL bzw. ihrer Erweiterungen.

c. dynamische und flexible P2P-Netze, (semi-)automatische Mapping-Tabellen und/oder Schema-Mediation, ferner die vernünftige Kombination zwischen verschieden Lösungsansätzen für vielfältige Bereiche der P2P-Systeme.

Diese Voraussetzungen führen zu einer erheblichen Komplexitätssteigerung der Datenintegration im P2P-Fall. Im Gegensatz dazu sieht man, dass in der existierenden P2P-Welt relativ einfache und nicht flexible Ansätze erfolgreich sind. Die Forscher haben die große Aufgabe, diese Kluft zwischen Theorie und Praxis im Bereich P2P-Datenintegration zu überwinden.

6. Literatur

[Ab+01] Aberer, Karl u. a.: *P-Grid: A Self-organizing Structured P2P System.* Aus: http://www.p-grid.org/Papers/CoopIS2001.pdf

[Ab+02] Aberer, Karl; Punceva, Magdalena; Hauswirth, Manfred; Schmidt, Roman: *Improving Data Access in P2P Systems.* Aus: http://www.p-grid.org/Papers/IC2002.pdf

[AH02] Aberer, Karl; Hauswirth, Manfred: *An Overview on Peer-to-Peer Information Systems*. Aus: http://gunther.smeal.psu.edu/18518.html

[Ar+03] Arenas, M., u. a.: *The Hyperion Project: From Data Integration to Data Coordination.* ACM SIGMOD Record, 32 (3), Sep. 2003.

[Au04] AutoMed: Automatic Generation of Mediator Tools for Heterogeneous Database Integration. http://www.doc.ic.ac.uk/automed/.

[Be+01] Berners-Lee, T.; Hendler, J. und Lassila, O.: *The semantic web*. *Scientific American*, May 2001.

[Be03] Berners-Lee, Tim: *Foreword*. In: *Spinning the Semantic Web*. Fensel, Dieter; Hendler James; Lieberman, Henry; Wahlster, Wolfgang (eds.), MIT Press, 2003, S. xi-xxiii.

[Ber+01] Bernstein, A. Philip u. a.: *Data Management for Peer-to-Peer Computing: A Vision.* Aus: http://www.db.ucsd.edu/webdb2002/papers/15.pdf

[BK+03] Busse, Susanne; Kabisch, Thomas; Leicher, Andreas; Löser, Alexander: Mediatorbasierte Integration populärer WWW-Services und –Portale in eine Peer-To-Peer Infrastruktur. Aus: http://cis.cs.tu-berlin.de/~aloeser/publications/XMIDX_Workshop_2003_P2P_Mediator_CIS_TU_Berlin.pdf

[FH+03] Fensel, Dieter; Hendler, Jim; Liebermann, Henry; Wahlster, Wolfgang: *Introduction*. In: *Spinning the Semantic Web*. Wahlster, Wolfgang (eds.), MIT Press, 2003, S. 1-25.

[Ge04] The Genome Database Website. http://www.gdb.org/.

[Gn04] Gnutella Website. http://www.gnutella.com/.

[GP+97] Garcia-Molina, Hector; Papakonstantinou, Yannis; Quass, Dallan; Rajaraman, Anand; Sagiv, Yehoshua; Ullman, Jeffrey; Vassalos, Vasilis; Widom, Jennifer: *The TSIMMIS Approach to Mediation: Data Models and Languages.* Journal of Intelligent Systems, Volume 8, Number 2, S. 117-132, März/April 1997. Aus: http://www-db.stanford.edu/pub/papers/tsimmis.ps

[Ha+01] Halevy, Alon u. a.: *Answering queries using views: A survey.* VLDB Journal, 10(4), 2001, S. 270/294.

[Ha+03a] Halevy, A., u. a.: *The Piazza Peer Data Management System.* In: IEEE Trans. on Knowledge and Data Engineering, 2003.

[Ha+03b] Halevy, A.; Ives, Z.; Mork, P. und Tatarinov, I.: *Piazza: Data Management Infrastructure for Semantic Web Applications.* In *WWW*, 2003.

[Ha+03c] Halevy, A.; u. a.: *Schema Mediation in Peer Data Management Systems*, ICDE 2003.

[Ha+03d] Halevy, Alon u. a.: *Piazza: Data Management Infrastructure for Semantic Web Applications.* Aus: http://www.cis.upenn.edu/~zives/research/piazza-www03.pdf

[Ha+03e] Halevy, Alon u. a.: *Crossing the Structure Chasm.* Aus: http://data.cs.washington.edu/homes/Pubs/ChasmCIDRo3.pdf

[Hy04] The Hyperion Project. http://www.cs.toronto.edu/db/hyperion/.

[KA+03a] Kementsietsidis, A.; Arenas, M.; Miller, R.: *Managing Data Mappings in the Hyperion Project.* Proc. of the Int. Conf. on Data Engineering (ICDE) 2003.

[KA+03b] Kementsietsidis, A.; Arenas, M.; Miller, R.: *Mapping Data in Peer-to-Peer Systems: Semantics and Algorithmic*. Issues. SIGMOD 2003.

[MB+02] Madhavan, J.; Bernstein, P.; Domingos, P.; Halevy, A.: *Representing and Reasoning about Mappings Between Domain Models.* Proc. of the AAAI Eighteenth National Conference on Artificial Intelligence 2002.

[MP03a] McBrien, P.; Poulovassilis, A.: *Data Integration by Bi-Directional Schema Transformation Rules.* Aus: http://www.doc.ic.ac.uk/~pjm/research/MP03a.ps

[MP03b] McBrien, P.; Poulovassilis, A.: *Defining peer-to-peer data integration using both as view rules.* Intern. Workshop On Databases, Information Systems and Peer-to-Peer Computing, Berlin 2003.

[Na04] Napster Website. http://www.napster.com/.

[Ne+01] Nejdl, Wolfgang, u.a., *Edutella: A p2p networking infrastructure based on rdf*, 2001. Aus: http://edutella.jxta.org/reports/edutella-whitepaper.pdf.

[On04] Online Mendelian Inheritance in Man. http://www.ncbi.nlm.nih.gov/omim/.

[Pg04] P-Grid Website. http://www.p-grid.org/.

[Pi04] Piazza Website. http://data.cs.washington.edu/p2p/piazza/.

[Po03] Poulovassilis, Alexandra: *AutoMed: Automatic generation of Mediator tools for heterogeneous data integration.* Aus: http://www.dcs.bbk.ac.uk/~ap/talks/abdn2003AutoMedPres.ppt

[Ra02] Rahm, Erhard; *Data Warehousing und Data Mining*. Aus: http://dsb.uni-leipzig.de/de/lehre/db-lernmaterial-vorl.html , SS 2002.

[Sa04] Sagres Website. http://data.cs.washington.edu/ubiquitous/sagres/.

[SG+01] Serafini, Luciano; Giunchiglia, Fausto; Mylopoulos, John; Bernstein, Philip A.: *The Local Relational Model: Model and Proof Theory.* Aus: http://sra.itc.it/people/serafini/distribution/0112-23.pdf

[Sw04] The SWISS-PROT Protein Knowldgebase. http://www.ebi.ac.uk/swissprot/.

[Ta03] Tatarinov, I.; u. a.: *The Piazza Peer Data Management Project.* ACM SIGMOD Record, 32 (3), Sep. 2003.

[Tz+03] Tzitzikas u. a.: *Taxonomy-based Conceptual Modeling for Peer-to-Peer Networks*. ER, 2003.

[Vo99] Vossen, Gottfried: *Datenmodelle, Datenbanksprachen und Datenbank-Management-Systeme*, Oldenbourg, 3. Auflage, 1999.