Stefan Conrad

Integration heterogener Datenbestände[1]

Dieser Beitrag gibt einen kurzen Überblick über die Problematik der Integration heterogener Datenbestände (Datenbanken und anderer Datenquellen). Darüber hinaus wird der Stand der Forschung in diesem Bereich vorgestellt und diskutiert. Da an dieser Stelle die detaillierte Behandlung einzelner Verfahren nicht möglich ist, wird darauf verzichtet. Für eine intensivere Beschäftigung mit einzelnen Verfahren sei auf die Literatur verwiesen.

Motivation

Die Zusammenführung (Integration) von Datenbeständen aus mehreren, typischerweise unterschiedlichen Quellen ist eine ständig in der Praxis auftretende Aufgabenstellung. Diese Aufgabe mag sich je nach konkretem Anwendungskontext in verschiedenen Ausprägungen zeigen, die prinzipiellen Fragestellungen sind aber immer dieselben. Die zentralen Probleme entstehen aus der Heterogenität der Datenquellen sowie aus der Tatsache, dass einerseits die Semantik der einzelnen Datenquellen vielfach überhaupt nicht exakt nachweisbar ist und andererseits die Beziehungen der Daten aus den verschiedenen Datenquellen untereinander oft nicht oder nur unvollständig bekannt sind.

Insbesondere im Bereich der Datenbankintegration werden die dabei auftretenden Probleme seit über 25 Jahren in der Forschung behandelt und Lösungen entwickelt. Datenbankintegration bezieht sich üblicherweise auf die Integration zweier oder mehrerer Datenbanken, wobei es sich hierbei um Datenbanken im engeren Sinne handelt, d. h., die Daten werden von einem richtigen Datenbankmanagementsystem verwaltet. Den Daten liegt jeweils eine Beschreibung in Form eines Datenbankschemas zugrunde. Abhängig vom Datenbankmanagementsystem erfolgt diese Datenbeschreibung z. B. auf der Basis des heute weit verbreiteten relationalen Datenmodells, in einem konzeptionellen Datenmodell wie dem *Entity-Relationship*-Modell oder in einem objektorientierten Modell. Zu diesem „klassischen“ Bild der Datenbankintegration sind in den letzten Jahren durch neue Anwendungen weitere Aspekte hinzugekommen. Von besonderem Interesse ist gegenwärtig die Einbeziehung von so genannten semistrukturierten Daten, wie sie häufig im World Wide Web zu finden sind. Auch der sich im Bereich der Standardisierung und der Umsetzung in kommerzielle Produkte abzeichnende Erfolgszug von XML (*Extensible Markup Language*), das als Basis zur Beschreibung, Strukturierung und letztendlich auch Speicherung von Dokumenten jeglicher Art eingesetzt wird (oder werden soll), zeigt die praktische Notwendigkeit.

Betrachtet man die heutzutage vorherrschenden Anwendungsfälle, in denen eine Daten(bank)integration gewünscht wird oder unabdingbar ist, so kann man grob zwei große Bereiche unterscheiden, wobei die Grenzen zunehmend verwischen.

[1] Ein Teil dieses Textes basiert auf Arbeiten, die im Rahmen des von der Deutschen Forschungsgemeinschaft (DFG) unterstützten Projekts „DIAsDEM: Datenintegration von Altlasten und semistrukturierten Dokumenten durch Einsatz von Mining Verfahren“ (CO 207/13-1) durchgeführt wurden (und fortgesetzt werden).

Der eine Bereich umfasst Integrationsaufgaben im betrieblichen Umfeld. Dieser Bereich kann als der klassische bezeichnet werden, in dem schon seit langer Zeit bestehende betriebliche Informationssysteme in der einen oder anderen Weise integriert oder verbunden werden sollen, um eine gemeinsame Nutzung der von diesen Systemen verwalteten Datenbanken (bzw. Datenbeständen) möglichst durch Schaffung eines einheitlichen Zugriffs auf die Daten zu ermöglichen, wobei die Frage der Daten(bank)integration hierbei als Teil des aktuellen Schlagworts *Enterprise Application Integration* (EAI) gerade wieder neu entdeckt wird. Die seit einiger Zeit sehr populär gewordenen Fusionen von Unternehmen verschärfen die Anforderungen hinsichtlich der Integration im betrieblichen Bereich deutlich, da nun häufig nicht nur einzelne Systeme integriert werden sollen, sondern gleich ganze IT-Systemlandschaften der fusionierenden Unternehmen. Auch wenn es von den Verantwortlichen öffentlich nicht thematisiert wird, ist dies mit einer der entscheidenden Punkte, weswegen viele Fusionen nicht (oder nicht so schnell) den erwarteten Erfolg bringen.

Der andere (in sich heterogenere) Anwendungsbereich zeichnet sich dadurch aus, dass auf Daten aus verschiedenen Quellen, oft aus dem World Wide Web, zugegriffen werden soll, die miteinander und gegebenenfalls mit eigenen Daten verknüpft werden sollen. Besonderes Charakteristikum ist hier, dass die verwendeten Datenquellen zumeist ein Informationsangebot von anderen Personen oder Institutionen darstellen, wobei diese Anbieter oft nicht wissen, wer diese Daten wofür nutzt. Dies hat aber auch z. B. zur Folge, dass die Daten in einer für die jeweiligen Anwendungszwecke nicht direkt geeigneten Repräsentation vorliegen können und dass benötigte Informationen zur Semantik der Daten nicht vollständig zugänglich sind. Als ein solches spezielles Anwendungsszenario können etwa Bereiche der Bioinformatik angeführt werden, in denen die Nutzung vieler und oft sehr heterogener Datensammlungen, die im World Wide Web von vielen Forschungseinrichtungen weltweit angeboten werden, unabdingbar ist.

Häufig wird noch zwischen *physischer* und *logischer* Integration unterschieden. Bei einer physischen Integration werden die Daten kopiert und zusammen in einer neuen Datenbank abgelegt. Ein typisches Beispiel für diesen Ansatz sind *Data Warehouses*.[2] Bei einer logischen Integration bleiben die Daten in ihren ursprünglichen Quellen und werden nur bei Anfragen an den integrierten Datenbestand – soweit zur Beantwortung der Anfragen notwendig – zusammengeführt; eine dauerhafte Speicherung der integrierten Daten ist dabei nicht vorgesehen. Diese Art der Integration findet in reiner Form in so genannten *föderierten Datenbanksystemen*[3] ihren Ausdruck; in der Praxis finden sich vielfältige Systemarchitekturen, die auf dem Prinzip der logischen Integration basieren.

Hinsichtlich der bei der Integration zu berücksichtigenden und zu lösenden Probleme sind die Unterschiede zwischen physischer und logischer Integration fast vernachlässigbar. Die Wahl zwischen diesen beiden Integrationsformen wird vor allem durch den Aufwand beim späteren Betrieb des integrierten Systems bestimmt.

Im Folgenden gehen wir zunächst kurz auf eine grobe Systemarchitektur sowie auf eine Schemaarchitektur ein. Anschließend erläutern wir den Integrationsprozess, bevor wir die bei der Integration zu lösenden Probleme (die so genannten Integrationskonflikte) im Detail thematisieren. Danach stellen wir einige wesentliche Beschränkungen bestehen-

[2] Vgl. Bauer und Günzel (2001) und Lehner (2003).
[3] Vgl. Sheth und Larson (1990) und Conrad (1997).

der Integrationsansätze vor und schließen mit einer – sicher unvollständigen – Aufzählung aktueller Fragestellungen und Herausforderungen an die Forschung in diesem Bereich. In diesem Beitrag gehen wir nicht im Detail auf konkrete Verfahren ein; dies würde den Rahmen sprengen. Stattdessen sei hier auf entsprechende Übersichten, wie etwa Conrad (1997) und Conrad (2002), sowie die an entsprechender Stelle aufgeführte Literatur verwiesen.

Architektur föderierter Systeme

Bei der Daten(bank)integration werden vorhandene Datenquellen oder Datenbanksysteme (mit ihren jeweiligen lokalen Anwendungen) in ein neues, umfassendes System eingebunden, das den Benutzern einen einheitlichen Zugriff auf die verschiedenen lokalen, oft heterogenen Datenbestände ermöglicht. Dabei sind aber die bestehenden Systeme weitestgehend zu erhalten, damit (lokale) Anwendungen, die für diese Systeme entwickelt wurden, auch weiterhin ohne Veränderung eingesetzt werden können.

Abhängig davon, welches Maß an Datenbanksystem-Funktionalität dieses Gesamtsystem bieten soll, gibt es verschiedene Systemarchitekturen, die es unterstützen. Der wohl weitestgehende Ansatz stellt den Idealfall dar, dass das neue Gesamtsystem sich den globalen Benutzern gegenüber wie ein einzelnes homogenes Datenbanksystem verhält. Diese Form der Datenbankintegration wird *Föderiertes Datenbanksystem* genannt.[4] Auch wenn in vielen Anwendungsbereichen das Gesamtsystem nicht volle Datenbanksystem-Funktionalität bieten muss, bleibt das Problem der Schema- und Datenintegration bestehen.

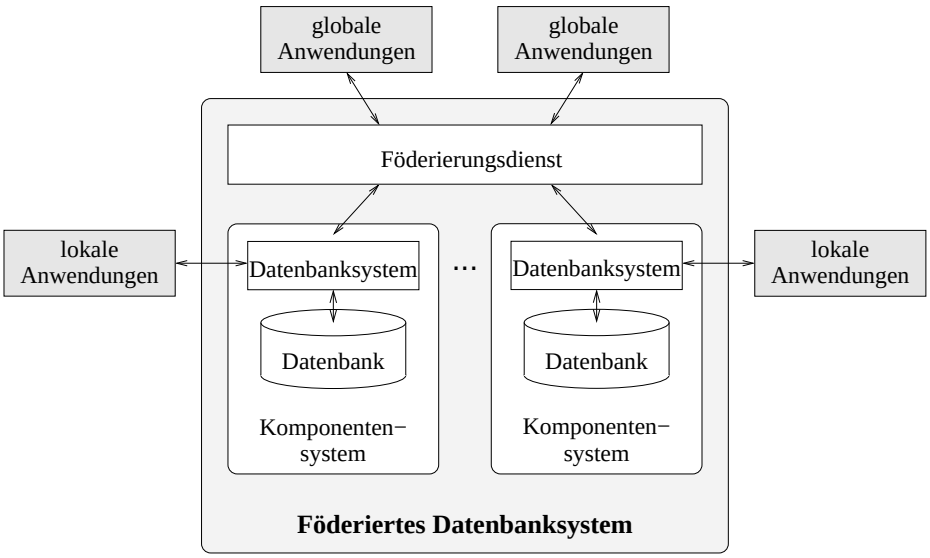

Abb. 1: Systemarchitektur föderierter Datenbanksysteme (hier mit zwei Komponentensystemen)

Abbildung 1 skizziert den Zusammenschluss bestehender Datenbanksysteme zu einem föderierten Datenbanksystem. Die bestehenden Datenbanksysteme gehen mit ihren Datenbeständen als *Komponentensysteme* in das föderierte System ein. Eine integrierende Komponente (*Föderierungsdienst*) ermöglicht neuen globalen Anwendungen den *einheit-*

[4] Vgl. u. a. Heimbigner und McLeod (1985), Sheth und Larson (1990) und Conrad (1997), wobei die Terminologie in der Literatur nicht ganz einheitlich ist.

lichen Zugriff auf den integrierten Gesamtdatenbestand. Die bereits bestehenden lokalen Anwendungen können weiterhin direkt über das entsprechende Komponentensystem auf den von ihnen benötigten Datenbestand zugreifen.

In der Regel werden die Daten in einem föderierten System nur *logisch* integriert. In speziellen Anwendungsfeldern kann es sinnvoll sein, die logisch integrierten Daten auch *physisch* integriert zur Verfügung zu stellen (in Form so genannter *materialisierter Sichten*[5]). Auch beim Aufbau eines *Data Warehouse*[6] können die Integrationskonzepte von Datenbank-Föderationen genutzt werden, um die Daten aus verschiedenen Quellsystemen in das *Data Warehouse* in ein einheitliches Format zu kopieren und dann darauf aufwändigere betriebswirtschaftliche Auswertungen durchzuführen, die nicht den Betrieb der anderen Systeme behindern sollen.

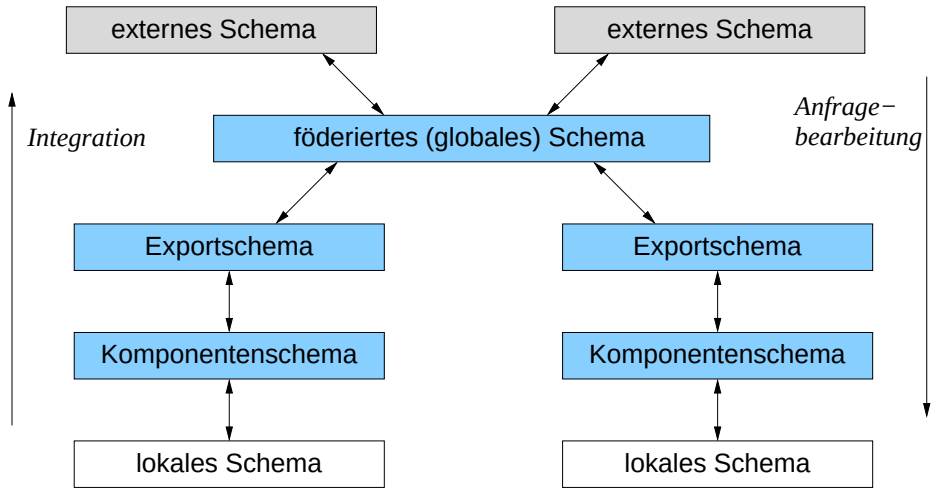

Abb. 2: *5-Ebenen-Schema-Architektur* nach Sheth und Larson (1990).

Als Referenzarchitektur wird in der Literatur zumeist die *5-Ebenen-Schema-Architektur* nach Sheth und Larson (1990) verwendet, die in Abbildung 2 dargestellt ist. Auch wenn diese Architektur aus verschiedenen Gründen in der Praxis so nicht umgesetzt wird (zumindest nicht vollständig), ist sie gerade zum Vergleich verschiedener Lösungen sehr gut geeignet.

Diese Architektur betrachtet fünf Schemaebenen. Für eine klassische Integration beginnt man üblicherweise bei den *lokalen Schemata* der zu integrierenden Komponentensysteme. Diese beschreiben den jeweils verwalteten Datenbestand mit den Mitteln des jeweiligen Datenmodells der Komponentensysteme. Um die Heterogenität dieser Datenmodelle zu überwinden, wird eine Transformation der lokalen Schemata in *Komponentenschemata* durchgeführt. Hierbei handelt es sich um eine Datenmodelltransformation, da die Komponentenschemata in dem globalen Datenmodell ausgedrückt werden, das für die Integration gewählt wurde. Auf der nächsten Ebene wird durch *Exportschemata* angegeben, welche Teile des jeweiligen lokalen Datenbestands in die Integration einfließen und damit später globalen Anwendungen zugänglich sein sollen. Die Bildung eines *föderierten Schemas* (oft auch *globales Schema* genannt) über den zur Verfügung gestellten

[5] Vgl. Staudt und Jarke (1996).
[6] Vgl. Bauer und Günzel (2001) und Lehner (2003).

Exportschemata schafft dann eine Beschreibung des integrierten Datenbestands. Komponentenschemata, Exportschemata und das föderierte Schema sind in dem gewählten globalen Datenmodell ausgedrückt. Als fünfte Ebene können noch *externe Schemata* definiert werden, die die Schnittstelle für globale Anwendungen darstellen, die auf den integrierten Datenbestand zugreifen. Solche externen Schemata gibt es prinzipiell in jeder Art von Datenbanksystemen. Sie erlauben eine anwendungsspezifische Sicht auf den Datenbestand und sind gleichzeitig Teil eines Schutzmechanismus, um Zugriffe auf andere Daten zu verhindern.

Integrationsprozess

Die Forschung im Bereich der Datenbankintegration hat eine Vielzahl von Techniken und Verfahren hervorgebracht, die es erlauben, verschiedene Aspekte der Integration zu behandeln. Die Vielzahl der zur Verfügung stehenden einzelnen Verfahren darf aber nicht darüber hinwegtäuschen, dass es zum einen bis jetzt keine wirklich einheitliche und durchgängige Methodik für die Durchführung der Integration auf der Basis solcher Verfahren gibt, und dass zum anderen die Anwendbarkeit vieler, wenn nicht sogar fast aller entwickelten Verfahren sehr eingeschränkt ist, da sie von Voraussetzungen ausgehen, die in vielen konkreten Anwendungsbereichen so nicht bestehen. Ein typisches Beispiel dafür ist, dass davon ausgegangen wird, dass die Semantik einer Datenbank vollständig bekannt und in für das jeweilige angewendete Verfahren verarbeitbarer Form beschrieben ist. In der Praxis muss man aber immer wieder feststellen, dass sich selbst die Anwendungsexperten z. B. in einem Unternehmen hinsichtlich der genauen Semantik der Daten in ihren Systemen nicht einig sind. Die Ursachen dafür können vielfältig sein. So kann bei Informationssystemen, die schon sehr lange im Einsatz sind, d. h., deren ursprünglicher Entwurf bereits viele Jahre zurückliegt und an denen seitdem viele kleine Änderungen vorgenommen wurden, die ein einzelner Anwender oder Fachexperte nicht mehr überblickt, das Wissen über die Semantik des ursprünglichen Entwurfs des Systems inzwischen verloren gegangen sein, weil die entsprechenden Mitarbeiter das Unternehmen schon vor langer Zeit verlassen haben.

Die Anwendung einzelner Verfahren beschränkt sich immer auf einen oder wenige Schritte in einem *Integrationsprozess*. Man kann den Integrationsprozess wie folgt in fünf Phasen aufteilen:

1. Analyse der zu integrierenden Datenquellen mit einer möglichst vollständigen Bestimmung der Semantik der zu integrierenden Daten,
2. Transformation der gegebenenfalls heterogenen Beschreibungen der Daten (Datenbankschemata) in ein gemeinsames Datenmodell,
3. Feststellung der sich semantisch entsprechenden Daten auf Schemaebene (durch Angabe so genannter Korrespondenzen zwischen den Elementen mehrerer Datenbankschemata),
4. Ableitung eines integrierten Schemas unter Verwendung der zuvor bestimmten Korrespondenzen,
5. Integration der Daten (bei physischer Integration Füllen der neuen Datenbank, für die in Phase 4 das Datenbankschema entstanden ist; bei logischer Integration Entwicklung

und Bereitstellung von Mechanismen, die bei der Anfragebearbeitung die eigentliche Integration der Daten vollziehen können).

Innerhalb dieser fünf Phasen können dann verschiedene Verfahren zum Einsatz kommen. Die Wissenschaft hat sich um diese fünf Phasen mit sehr unterschiedlicher Intensität gekümmert. Sehr lange wurden hauptsächlich Verfahren für die Phasen 4 und 2 erarbeitet. Seit einigen Jahren werden auch für die Phase 5, insbesondere im Bereich der Anfragebearbeitung, neue Techniken entwickelt. Die Phasen 1 und 3 wurden sehr lange vernachlässigt und finden erst seit kurzer Zeit das Interesse der Forschung. Die Bedeutung der Phasen 1 und 3 für eine erfolgreiche Integration zeigt sich in den Problemen, die in der praktischen Durchführung größerer Integrationsprojekte regelmäßig auftreten:

- Die Datenbankschemata (von Datenbanken, die von Datenbankmanagementsystemen verwaltet werden) sind oft nicht vollständig, d. h., es sind nicht alle für die Integration benötigten semantischen Informationen explizit im Datenbankschema vorhanden.
- Die Datenquellen sind oft „semistrukturiert“, oder es gibt überhaupt kein Datenbankschema.
- In Altsystemen (*legacy systems*) ist die tatsächliche Semantik der Daten in der Datenbank in der Regel nicht vollständig bekannt.
- Die Korrespondenzen und Abhängigkeiten zwischen Daten aus verschiedenen Quellen sind nicht (vollständig) bekannt. Häufig können Fachexperten nur vermuten, welche Korrespondenzen korrekt sind.

Neuere Arbeiten versuchen, diese Lücken zu schließen. Der Einsatz von Datenanalyseverfahren verspricht hier viel Erfolg.[7] Speziell die Entwicklungen aus dem Bereich *Knowledge Discovery in Databases* (KDD) kommen für diesen Zweck in Frage. Diese auch *Data Mining* genannten Verfahren sollen fehlende semantische Informationen aus vorhandenen Datenbeständen ableiten. Auch für die Bestimmung von Schemakorrespondenzen können sie eingesetzt werden. Da *Data Mining*-Verfahren zunächst nicht speziell für die Nutzung zur Unterstützung der Datenbankintegration entwickelt wurden, ist gegebenenfalls eine Anpassung oder spezifische Weiterentwicklung erforderlich.

Als Beispiele möglicher Anwendungen von *Data Mining*-Verfahren im Datenbank-Integrationsprozess seien hier genannt:

- Verfahren zum Finden so genannter Assoziationsregeln, die eventuell nicht explizit formulierte Integritätsbedingungen darstellen und damit bislang fehlende semantische Information zu einem Datenbankschema liefern können;
- Verfahren zum Entdecken von Strukturinformation sowie zum *Clustering* und zur Klassifikation, mit denen bei semistrukturierten Datenquellen eine einem Datenbankschema vergleichbare Strukturierung und Zuordnung der Daten in diese Struktur ermöglicht wird;
- Verfahren zum Datenvergleich, die gleiche (oder ähnliche) Objekte entdecken, zu denen Daten in mehreren Datenquellen vorliegen und die damit Vorschläge für Schemakorrespondenzen erzeugen können.

[7] Vgl. auch Höding und Conrad (1998).

Neben solchen *Data Mining*-Verfahren spielen in der Phase 3 insbesondere so genannte *Schema Matching*-Verfahren eine große Rolle. Wie der Name bereits andeutet, versuchen diese Verfahren, bestehende Schemata miteinander zu vergleichen und eine Abbildung (*mapping*) zu bestimmen, mit der einander entsprechende Schemaelemente zueinander in Beziehung gesetzt werden. Hierzu werden je nach Verfahren z. B. linguistische Beziehungen zwischen Bezeichnern in Schemata, strukturelle Ähnlichkeiten, Ähnlichkeiten der zugehörigen Daten und gegebenenfalls Kombinationen dieser Eigenschaften verwendet. Einen detaillierten Überblick über die verschiedenen Arten von *Schema Matching*-Verfahren geben Rahm und Bernstein (2001).

Der Einsatz von *Data Mining*- sowie von *Schema Matching*-Verfahren führt zwangsläufig zu der Frage, wie verlässlich die damit erzielten Informationen sind. Schon die Tatsache, dass diese Verfahren nur auf einzelnen, aktuell vorhandenen Datenbeständen angewendet werden, zeigt, dass die gewonnene Information mit einer gewissen Unsicherheit behaftet ist. Diese Unsicherheit ließe sich nur ausschließen, wenn alle prinzipiell möglichen Datenbankzustände analysiert würden. Dies ist im Allgemeinen nicht durchführbar, da die Zahl der möglichen Zustände fast immer unendlich ist. Auf der anderen Seite muss man aber ebenso feststellen, dass auch die nicht mit solchen Verfahren gewonnenen Informationen (z. B. Informationen zur Semantik, die von Experten stammen) sowie die vorhandenen Daten selbst mit Unsicherheit behaftet sind. Die Qualität von Daten, die man in betrieblichen oder sonstigen Datenbanken findet, kann sehr unterschiedlich sein. Tipp- und andere Eingabefehler sowie Verarbeitungsfehler durch fehlerhafte Anwendungssoftware oder durch falschen Umgang mit Anwendungsprogrammen und Daten beeinträchtigen die Datenqualität und damit letztendlich auch alles, was von diesen Daten abhängig ist. Genauso sind aber auch Datenbankschemata nicht *per se* perfekt, da Fehler beim Datenbankentwurf sowie eine im Laufe der Zeit veränderte Nutzung der Datenbank dazu führen können, dass ein vorliegendes Datenbankschema nicht nur semantisch unvollständig ist, sondern auch semantisch falsche Informationen liefern kann, die damit zu falschen Folgerungen in den weiteren Integrationsphasen führen können.

Praktisch alle für die Datenbankintegration entwickelten Verfahren ignorieren diesen wichtigen Aspekt und setzen implizit perfektes und vollständiges Wissen voraus. Auf der Basis dieser Annahme liefern sie wiederum perfekte und (zumeist) eindeutige Ergebnisse. Da die Eingangsdaten und Informationen bei einer Datenbankintegration schon prinzipiell unsicher sind, wird diese Unsicherheit durch die verschiedenen zum Einsatz kommenden Integrationsverfahren weitergereicht; einige Verfahren fügen weitere unsichere Informationen hinzu, so dass im Laufe des Integrationsprozesses eine Aggregation der Unsicherheit stattfindet. Am Ende des Prozesses steht ein Integrationsergebnis, dessen Korrektheit vollständig in Frage gestellt werden muss, wenn wir keine Möglichkeiten zur Analyse der Unsicherheit der verarbeiteten Informationen und ihrer Auswirkung auf die Integration haben. In aktuellen Arbeiten haben wir daher die Ursachen für unsichere Informationen[8] untersucht und eine auf etablierten statistischen Verfahren basierende Methode zur Bewertung der Unsicherheit im Datenbank-Integrationsprozess entwickelt.[9]

[8] Vgl. Altareva und Conrad (2001).
[9] Vgl. Altareva und Conrad (2003).

Integrationskonflikte

Bei der Integration heterogener Datenbanken tritt – auch ohne die zuvor diskutierte Unsicherheit der verwendeten Informationen – eine Reihe von Problemen auf. Für den späteren Betrieb eines integrierten Informationssystems spielt beispielsweise eine Vielzahl technischer Fragen eine große Rolle, u. a. hinsichtlich der Kommunikation zwischen verschiedenen existierenden Informationssystemen. Dieser Bereich erfordert die Betrachtung diverser technischer Details, auf die wir hier nicht eingehen wollen. Stattdessen wollen wir uns auf die die semantischen Aspekte der Integration betreffenden Problembereiche konzentrieren. Hier spricht man in der Literatur auch von *Integrationskonflikten*, die im Integrationsprozess gelöst werden müssen. Wir unterscheiden dabei drei Ebenen, in denen solche Konflikte auftreten:

- heterogene Datenmodelle (Datenmodellebene),
- unterschiedliche Modellierung (Modellierungs- oder Schemaebene) und
- heterogene Repräsentation der Daten (Daten- oder Instanzebene).

Eine in der Literatur häufig zitierte Klassifikation von Integrationskonflikten unterscheidet semantische Konflikte, Beschreibungskonflikte, Heterogenitätskonflikte und strukturelle Konflikte.[10] Da diese Einteilung nicht alle relevanten Integrationskonflikte erfasst, orientieren wir uns im Folgenden an der Einteilung in die drei oben genannten Ebenen.

Datenmodellebene:

Die bei einer Integration zu berücksichtigenden Datenbanken und sonstigen Datenquellen verwenden oft unterschiedliche Datenmodelle bzw. Datenmodellen entsprechende Beschreibungskonzepte für die Daten. So liegt relationalen Datenbanken das Relationenmodell zugrunde, das die Daten in Tabellen (mathematisch: Relationen) organisiert. Objektorientierte Datenbanken verwenden eigene Objektmodelle zur Modellierung. In Altsystemen kommen zudem noch das hierarchische Datenmodell oder das Netzwerkmodell bei Datenbanken vor. Andere Datenquellen basieren implizit oder explizit auf einem semistrukturierten Datenmodell, wie es z. B. auch XML zugrunde liegt.

Problematisch an der Verwendung unterschiedlicher Datenmodelle ist, dass sie verschiedene Modellierungskonzepte anbieten und auch unterschiedliche Ausdrucksfähigkeit besitzen. Damit sind die in verschiedenen Datenmodellen ausgedrückten Datenbankschemata nur schwer direkt miteinander vergleichbar. Für die Integration bietet es sich daher an, die Modellierungen aus den verschiedenen Datenmodellen in ein einheitliches Datenmodell zu überführen, das als Basis für den ganzen Integrationsprozess dient.

Die Diskussion, welches Datenmodell dafür am besten geeignet ist, nimmt manchmal kuriose Züge an. Prinzipiell muss man sich dazu vor Augen führen, dass ein Datenmodell mit wenigen Modellierungskonzepten – wie das Relationenmodell – in der weiteren Verarbeitung sehr einfach ist, dafür aber die semantischen Feinheiten, die andere Datenmodelle unterscheiden können, nicht adäquat nachbilden kann. Verwendet man ein solches Datenmodell als Basis für die weitere Integration, gehen gegebenenfalls viele semantische Informationen verloren. Umgekehrt bedeutet die Verwendung eines Datenmodells, das mehr

[10] Vgl. Spaccapietra *et al.* (1992).

Modellierungskonzepte anbietet, dass man bei der Transformation in dieses Datenmodell eventuell fehlenden Informationen (die sich im Ursprungsmodell nicht repräsentieren ließen) hinzufügen muss, um eine einheitliche Modellierungsqualität zu erreichen. Es stellt sich dann aber oft die Frage, woher diese fehlenden Informationen beschafft werden können.

Die Transformation in ein als Basis der Integration gewähltes Datenmodell kann zu zusätzlichen Integrationskonflikten auf der Schemaebene führen.

Schemaebene:

Liegen zwei (oder mehr) Datenbankschemata, die integriert werden sollen, im gleichen Datenmodell vor, kann trotzdem eine Vielzahl von Integrationskonflikten auftreten. Ursache dafür ist im Wesentlichen, dass die Datenbanken in der Regel unabhängig voneinander entstanden sind und damit auch ihre Datenbankschemata unabhängig voneinander entworfen wurden. Die Freiheiten, die man in allen Datenmodellen bei der Modellierung der in der Datenbank zu repräsentierenden Sachverhalte hat, führt häufig zu unterschiedlichen Modellierungen der gleichen Sachverhalte, wodurch man unterschiedliche Strukturen in den Datenbankschemata erhält. Schon in einem einfachen Modell wie dem Relationenmodell lässt sich ein und derselbe Sachverhalt oft in einer oder in zwei Tabellen beschreiben, woraus unterschiedliche Strukturen im Schema entstehen.

Ein anderer Integrationskonflikt ergib sich aus der Tatsache, dass Datenbankplaner in der Wahl von Bezeichnern frei sind: für eine Datenbank mag z. B. Arbeitnehmer als Bezeichner gewählt worden sein, für eine andere Mitarbeiter. Für solche Bezeichner müssen wir bei der Integration immer mit Homonymen und Synonymen, aber auch z. B. mit Hyperonymen, rechnen, die sich im Allgemeinen einer automatisierten Auflösung entziehen.

Für die in den Datenbanken vorkommenden Objekte können dann noch unterschiedliche Eigenschaften vorgesehen sein, die gespeichert werden. Welche Eigenschaften in einer Datenbank gespeichert werden, hängt zumeist von den Anforderungen ab, die die Anwendungen stellen, für die eine Datenbank entworfen wird. So ist es für eine Anwendung im Bereich der Lohnbuchhaltung sicherlich erforderlich, das Gehalt der Arbeitnehmerinnen und Arbeitnehmer in der Datenbank zu haben, während für eine Anwendung zum Personaleinsatz eher die Qualifikationen und sonstigen Fähigkeiten der Mitarbeiterinnen und Mitarbeiter von Interesse sind.

Ein häufig auftretender Konflikt betrifft die Menge der jeweils gespeicherten (Daten-) Objekte. Die Lohnbuchhaltung erfasst wahrscheinlich alle Arbeitnehmerinnen und Arbeitnehmer im Unternehmen, die Personaleinsatzplanung vielleicht nur einen Teil des Unternehmens und damit nicht alle Arbeitnehmerinnen und Arbeitnehmer. Entsprechend werden sich vermutlich die Mengen der in den beiden Datenbanken repräsentierten Arbeitnehmerinnen und Arbeitnehmer unterscheiden.

Datenebene:

Datenkonflikte sind bei der Realisierung einer globalen Anfragebearbeitung (z. B. in einem föderierten Informationssystem) oder bei der Materialisierung integrierter Sichten (z. B. in einem *Data Warehouse*) von sehr großer Bedeutung. Typische Beispiele für Datenkonflikte sind unterschiedliche Konventionen bezüglich Schreibweisen von Namen

(„Meier, Petra“ und „Petra Meier“), Eingabe- und Tippfehler sowie die Verwendung unterschiedlicher Ausdrücke als Datenwerte (z. B. „Informatiker“ und „IT-Spezialist“ als Berufsbezeichnungen). Auch hier müssen wir also mit Homonymen, Synonymen und Hyperonymen rechnen.

Weitere häufig auftretende Datenkonflikte finden sich bei Zahlenwerten, wie etwa abgespeicherten Messwerten. Hier können z. B. *Skalierungskonflikte* auftreten, d. h., die Zahlenwerte haben unterschiedliche Maßeinheiten (z. B. 0,75 kg und 750 g). Auch können *Genauigkeitskonflikte* vorliegen, die einerseits in der Genauigkeit begründet liegen können, mit der die Daten in der Datenbank abgespeichert werden (in der Regel unterschiedlich viele Nachkommastellen), und die andererseits durch die Genauigkeit der Erfassung (Messung) entstehen können. Gerade, wenn die Messung eine deutlich geringere Genauigkeit hat, als die Daten in der Datenbank abgespeichert werden, entsteht später oft der falsche Eindruck von Daten mit höherer Genauigkeit. Bei der Integration muss man ferner noch auf *Vagheitskonflikte* achten: Numerische Daten können auch durch Schätzungen oder grobe Rundungen entstanden sein. Vagheiten können aber auch bei nichtnumerischen Daten auftreten. So sind Angaben wie „groß“ oder „lang“ prinzipiell vage. Oft kann man darüber streiten, ob ein Genauigkeits- oder ein Vagheitskonflikt vorliegt. Als ein Beispiel dafür lassen sich Noten anführen, mit denen Leistungen bewertet werden. Nimmt man etwa eine Notenskala an, die nur die Noten 1, 2, 3, 4 und 5 zulässt, so hat man offensichtlich ein Genauigkeitsproblem, wenn man dies mit Daten vergleicht, denen eine andere Notenskala (z. B. mit den Noten 1,0, 1,3, 1,7 usw.) zugrunde liegt. Andererseits sind Noten oft „vage“ – in dem Sinne, dass die zu ein und derselben Note führenden Leistungen oft nicht wirklich miteinander vergleichbar sind (was in noch größerem Maße dann für Durchschnittsnoten gilt, wie sie regelmäßig auf Zeugnissen zu finden sind).

Im Bereich *Data Warehouses* hat sich unter der Bezeichnung *Data Cleaning* ein eigenes Teilgebiet entwickelt, um solche Probleme zu behandeln, die beim Aufbau eines *Data Warehouse* entscheidend für dessen Datenqualität sind, zu behandeln. Einen Überblick über *Data Cleaning*-Verfahren geben Rahm und Do (2000).

Beschränkungen bestehender Techniken

Die bestehenden Schemaintegrationstechniken[11] setzen sich nicht vollständig mit allen Arten von Integrationskonflikten auseinander. Während extensionale Konflikte von allen Integrationstechniken ausführlich behandelt werden können, wird insbesondere die Auflösung von Heterogenitäts- und Beschreibungskonflikten kaum unterstützt. In der Regel wird vorausgesetzt, dass die zu integrierenden Schemata im gleichen (dem globalen) Datenmodell vorliegen und dass insbesondere Homonym- und Synonymprobleme auf Schemaebene nicht vorkommen bzw. vorab aufgelöst wurden. Da Datentypen, Wertebereiche und zugrunde gelegte Maßeinheiten für Attribute von den meisten Verfahren nicht betrachtet werden, können auch daraus resultierende Konflikte nicht abgarbeitet werden. Hinsichtlich der Behandlung struktureller Konflikte sind ebenfalls Einschränkungen festzustellen. Eine ausführliche Analyse findet sich in Conrad (1997).

Betrachtet man die fünf Phasen, die wir im Abschnitt „Integrationsprozess“ vorgestellt haben, so kann man feststellen, dass besonders die Phasen 1, 3 und 5 problematisch sind.

[11] Als typisches Beispiel sei hier nur auf Spaccapietra *et al.* (1992) verwiesen.

In den Phasen 2 und 4 können weitgehend automatisierte Verfahren eingesetzt werden – vorausgesetzt, die vorhergehenden Phasen haben „perfekte“ Ergebnisse produziert. Da dies in den meisten Fällen unrealistisch ist, sind auch die Ergebnisse der Phasen 2 und 4 jeweils zu überprüfen und der Einfluss eventueller Unsicherheiten festzustellen. Die Verfahren, die für die Phasen 1, 3 und 5 entwickelt oder aus anderen Bereichen dafür übernommen wurden, liefern in der Regel mit Unsicherheiten verbundene Ergebnisse.

Auch wenn man sich perfekte Verfahren wünschen würde – eine vollständige automatisierte Bearbeitung dieser Phasen scheitert grundsätzlich daran, dass hier immer die Semantik der zu integrierenden Daten ins Spiel kommt, die im Allgemeinen nicht automatisch aus den Daten und den üblichen Beschreibungen der Daten (z. B. Datenbankschemata) erschlossen werden kann, da sie immer ein möglichst umfassendes Hintergrundwissen über den Anwendungsbereich voraussetzt, in dem die Daten verwendet werden.

Zusammenfassung und aktuelle Fragestellungen

Zum Themenkomplex „Integration heterogener Datenbestände“ werden gegenwärtig Fragestellungen in verschiedenen Richtungen untersucht.

Einen Bereich stellt die Weiterentwicklung von Multidatenbanksprachen dar. Hierbei werden verschiedene Erweiterungen bekannter Datenbanksprachen entwickelt, durch die Konfliktlösungsmechanismen (für Beschreibungskonflikte, strukturelle Konflikte, aber auch Datenkonflikte) in diesen Sprachen verfügbar werden. Zwei sehr unterschiedliche Beispiele für solche Arbeiten an Multidatenbanksprachen sind FRAQL[12] und MQL[13]. Beide haben als Basis die relationale Datenbanksprache SQL, wobei in FRAQL unter anderem explizite Konfliktlösungsmechanismen über benutzerdefinierbare Funktionen integriert werden, während MQL neue, sehr mächtige Anfrageoperationen auf Multidatenbanken einführt – im Gegensatz zu SQL, das nur Operationen auf Relationen einer einzelnen Datenbank kennt. Anhand von MQL werden vorrangig theoretische Komplexitätsuntersuchungen für Multidatenbankanfragen durchgeführt, während FRAQL auch in einem Prototyp realisiert wurde, womit die experimentelle Untersuchung z. B. von speziellen Optimierungstechniken möglich ist.

Ein anderer Bereich der aktuellen Forschung ist die Weiterentwicklung bestehender Schemaintegrationsverfahren zu inkrementellen Verfahren. Da insbesondere die extensionalen Korrespondenzen zwischen den Daten verschiedener Datenbanken *a priori* nicht immer bekannt sind, werden inkrementelle Verfahren benötigt, so dass auch mit unvollständigem Wissen bereits ein integriertes Schema gebildet werden kann, das dann schrittweise verfeinert werden kann, sobald neue extensionale Korrespondenzen bekannt werden. Gleichzeitig ermöglichen es inkrementelle Verfahren, frühzeitig einen Eindruck von dem Ergebnis der Integration zu bekommen, womit auch Entscheidungen im Integrationsprozess (beispielsweise hinsichtlich extensionaler Korrespondenzen) validiert und gegebenenfalls revidiert werden können.

In eine andere Richtung zielen die schon genannten Verfahren des *Schema Matching*.[14] Hierbei sollen soweit wie möglich automatisch Schemakorrespondenzen bzw. potenzielle Korrespondenzen bestimmt werden.

[12] Vgl. Sattler *et al*. (2000) und Sattler *et al*. (2003biblio).
[13] Vgl. Wyss *et al*. (2001).
[14] Eine aktuelle Übersicht findet sich in Rahm und Bernstein (2001).

Die Berücksichtigung von Unsicherheiten bei der Datenbankintegration ist erst seit wenigen Jahren Gegenstand der Forschung. Altareva und Conrad (2001) schlagen erstmals eine Klassifikation der Unsicherheiten entsprechend ihren verschiedenen Ursprüngen vor. Darauf basierend wird in Altareva und Conrad (2003) unter Verwendung etablierter statistischer Verfahren ein methodischer Rahmen entwickelt, mit dem eine Berücksichtigung der Unsicherheiten im Integrationsprozess und eine darauf aufbauende Bewertung des Integrationsergebnisses möglich wird. Dies soll dazu genutzt werden, den Einsatz der vielen unterschiedlichen Verfahren in den einzelnen Phasen der Integration besser zu steuern. Die zum Einsatz kommenden statistischen Verfahren setzen allerdings auch eine quantitative Erfassung der Unsicherheiten voraus. Viele Verfahren, insbesondere *Data Mining*- und *Schema Matching*-Verfahren, liefern bereits entsprechende Informationen über die Sicherheit des von ihnen produzierten Ergebnisses (etwa als Wahrscheinlichkeitswerte oder durch Angabe von *support*- und *confidence*-Werten).

Während in früheren Forschungsarbeiten der Aspekt der Qualität keine wesentliche Rolle spielte, da man von perfekten und vollständigen Ausgangsinformationen ausging, mit denen dann auch ein perfektes Ergebnis erreicht werden konnte, ist in den letzten Jahren in verschiedenen Bereichen Qualität zum Forschungsgegenstand geworden. In vielen Fällen hat sich beim Einsatz in der Praxis beim Aufbau eines *Data Warehouse* oder bei der Integration vorhandener Informationssysteme herausgestellt, dass theoretisch wohlüberlegte Ansätze häufig doch qualitativ unbefriedigende Ergebnisse liefern, da sie von zumeist nicht erfüllten Voraussetzungen ausgehen. Dem Problem der Bewertung der Schema- und Datenqualität haben sich daher verschiedene Forschergruppen und -projekte angenommen.[15]

Weitere aktuelle Fragestellungen, die in verschiedenen Beiträgen auf speziellen Workshops[16] aufgeworfen und diskutiert wurden, betreffen unter anderem die Akquisition und Verwaltung von Metadaten sowie Probleme der Evolution der Systeme. Es geht dabei um die Frage: Wie wirken sich lokale Schemaänderungen auf das integrierte Schema aus?

Bibliographie

ALTAREVA, Evguenia und Stefan CONRAD. „The Problem of Uncertainty and Database Integration“, in: KUTSCHE *et al*. (2001), 92-99.

ALTAREVA, Evguenia und Stefan CONRAD. „Analyzing Uncertainties in the Database Integration Process by Means of Latent Class Analysis“, in: Anne JAMES, Stefan CONRAD und Wilhelm HASSELBRING (Hrsg.). *Engineering Federated Information Systems, Proceedings of the 5th Workshop EFIS 2003*. Berlin und Amsterdam 2003, 14-22

BAUER, Andreas und Holger GÜNZEL (Hrsg.). *Data-Warehouse-Systeme*. Heidelberg 2001.

CONRAD, Stefan. *Föderierte Datenbanksysteme: Konzepte der Datenintegration*. Berlin und Heidelberg 1997.

CONRAD, Stefan. „Schemaintegration – Integrationskonflikte, Lösungsansätze, aktuelle Herausforderungen“, *Informatik – Forschung & Entwicklung* 17 (2002), 101-111.

GERTZ, Michael und Ingo SCHMITT. „Data Integration Techniques based on Data Quality Aspects“,

[15] Siehe z. B. Gertz und Schmitt (1998) und Jarke *et al*. (1999).
[16] Siehe etwa Roantree *et al*. (2000) und Kutsche *et al*. (2001).

in: Ingo SCHMITT, Can TÜRKER, Eyk HILDEBRANDT und Michael HÖDING (Hrsg.). *FDBS 3. Workshop „Föderierte Datenbanken“, Magdeburg, Germany, 1998*. Aachen 1998, 1-19.

HEIMBIGNER, Dennis und Dennis MCLEOD. „A Federated Architecture for Information Management“, *ACM Transactions on Office Information Systems* 3 (1985), 253-278.

HÖDING, Michael und Stefan CONRAD. „Data-Mining Tasks in Federated Database Systems Design“, in: Tamer ÖZSU, Asuman DOGAC und Özgür ULUSOY (Hrsg.). *Issues and Applications of Database Technology (IADT '98), Proceedings of the 3rd World Conference on Integrated Design and Process Technology*. Bd. 2. Berlin 1998, 384-391.

JARKE, Matthias, Manfred JEUSFELD, Christoph QUIX und Panos VASSILIADIS. „Architecture and Quality in Data Warehouses: An Extended Repository Approach“, *Information Systems* 24 (1999), 229-253.

KUTSCHE, Ralf, Stefan CONRAD und Wilhelm HASSELBRING (Hrsg.). *Engineering Federated Information Systems, Proceedings of the 4th Workshop EFIS 2001*. Berlin und Amsterdam 2001.

LEHNER, Wolfgang. *Datenbanktechnologie für Data-Warehouse-Systeme: Konzepte und Methoden*. Heidelberg 2003.

RAHM, Erhard und Hong Hai DO. „Data Cleaning: Problems and Current Approaches“, *IEEE Data Engineering Bulletin* 23 (2000), 3-13.

RAHM, Erhard und Philip A. BERNSTEIN. „A Survey of Approaches to Automatic Schema Matching“, *The VLDB Journal* 10 (2001), 334-350.

ROANTREE, Mark, Wilhelm HASSELBRING und Stefan CONRAD (Hrsg.). *Engineering Federated Information Systems, Proceedings of the 3rd Workshop EFIS 2000*. Berlin und Amsterdam 2000.

SATTLER, Kai-Uwe, Gunter SAAKE und Stefan CONRAD. „Adding Conflict Resolution Features to a Query Language for Database Federations“, *Australian Journal of Information Systems* 8 (2000), 116-125.

SATTLER, Kai-Uwe, Stefan CONRAD und Gunter SAAKE. „Interactive Example-driven Integration and Reconciliation for Accessing Database Federations“, *Information Systems* 28 (2003), 393-414.

SHETH, Amit P. und James A. LARSON. „Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases“, *ACM Computing Surveys* 22 (1990), 183-236.

SPACCAPIETRA, Stefano, Christine PARENT und Yann DUPONT. „Model Independent Assertions for Integration of Heterogeneous Schemas“, *VLDB Journal* 1 (1992), 81-126.

STAUDT, Martin und Matthias JARKE. „Incremental Maintenance of Externally Materialized Views“, in: T. M. VIJAYARAMAN, A. P. BUCHMANN, C. MOHAN und N. L. SARDA (Hrsg.). *Proceedings of the 22nd International Conference On Very Large Data Bases, VLDB '96, Bombay*. San Francisco 1996, 75-86.

WYSS, Catharine, Felix WYSS und Dirk VAN GUCHT. „Augmenting SQL with Dynamic Restructuring to Support Interoperability in a Relational Federation“, in: KUTSCHE *et al*. 2001, 5-18.