Techniken des Data Merging in Integrationssystemen

Jens Bleiholder
Humboldt-Universität zu Berlin, Institut für Informatik
bleiho@informatik.hu-berlin.de

Zusammenfassung

Die Integration von Daten aus heterogenen Informationsquellen ist ein bekanntes Forschungsthema im Datenbankbereich. Bei der Integration treten drei verschiedene Arten der Heterogenität auf: Technische Heterogenität bezeichnet die Heterogenität auf Plattform- und Formatebene. Strukturelle Heterogenität bezeichnet die Heterogenität auf Schemaebene. Diese tritt auf, wenn gleiche Domänen unterschiedlich modelliert werden. In letzter Zeit tritt verstärkt semantische Heterogenität, die sich mit den Daten und deren Bedeutung befasst, als Forschungsthema in den Vordergrund.

Im Rahmen des HumMer-Systems zur Integration heterogener Informationsquellen wird semantische Heterogenität auf Datenebene betrachtet. Unterschiedliche Informationsquellen können sowohl sich ergänzende als auch widersprüchliche Daten über gleiche Objekte enthalten. Wie solche Konflikte in Daten automatisch und sinnvoll aufgelöst werden können, ist eine noch nicht zufriedenstellend beantwortete Fragestellung. Der Beitrag gibt einen Überblick über bereits existierende Verfahren des *data merging*, wie sie in ausgewählten integrierenden Informationssystemen (z.B. TSIMMIS) angewandt werden. Desweiteren werden Techniken zur Integration genannt, die auf relationaler Algebra basieren. Zuletzt wird die Architektur des integrierenden Informationssystems HumMer (Humboldt Merger) vorgestellt. Dabei wird auf den Entwurf und die Realisierung der Data Merging Komponente näher eingegangen.

1 Semantik des Data Merging

Integrationssysteme stellen Nutzern eine einheitliche Sicht auf mehrere Informationsquellen zur Verfügung. Anstatt viele Anfragen in verschiedenen Sprachen an unterschiedliche Schemata zu stellen, wird bei der Verwendung eines Integrationssystems nur eine Anfrage gestellt. Die Verteilung der Anfrage an die einzelnen Informationsquellen und das Zusammenführen der Informationen aus den Quellen übernimmt dabei das Integrationssystem. Data Merging bezeichnet im Folgenden den Prozess des Zusammenführens der Daten aus den einzelnen Quellen unter Auflösung von auftretenden Konflikten. Dabei wird vorrausgesetzt, dass bekannt ist, welche Objekte zusammengeführt werden, z.B. durch eine global eindeutige und konsistente ID.
Tabelle 1 zeigt als Beispiel zwei Relationen, die jeweils die Daten einer Informationsquelle repräsentieren. Beide Relationen beschreiben Personen. Sie überlappen sich sowohl horizontal in

Daten aus Quelle 1			
Name	*Alter*	*Student*	*Ort*
Felix	⊥	nein	Hamburg
Melanie	23	ja	⊥
Jens	⊥	ja	Karlsruhe
Christoph	25	ja	Berlin

Daten aus Quelle 2			
Name	*Alter*	*Student*	*Telefon*
Melanie	⊥	ja	030/12345
Jens	27	⊥	030/54321
Christoph	24	ja	⊥
Karsten	24	nein	030/98765

Ergebnis nach Data Merging von Quelle 1 und Quelle 2				
Name	*Alter*	*Student*	*Ort*	*Telefon*
Felix	⊥	nein	Hamburg	⊥
Melanie	23	ja	⊥	030/12345
Jens	27	ja	Karlsruhe	030/54321
Christoph	24	ja	Berlin	⊥
Karsten	24	nein	⊥	030/98765

Tabelle 1: Beispiele für Konflikte in 2 Quellen

den Attributen, als auch vertikal in den Personen. Desweiteren ergänzen sich die Relationen: das Attribut `Ort` ist nur in Relation 1 enthalten, Attribut `Telefon` ausschliesslich in Relation 2.
Wie leicht zu erkennen ist, ergeben sich auf Datenebene Konflikte. So sind z.B. die Personen *Melanie*, *Jens* und *Christoph* in beiden Quellen beschrieben. Es sind zwei verschiedene Konfliktarten zu unterscheiden: a) „Unsicherheiten“, also fehlende Information durch `NULL`-Werte ($\perp$) in den Relationen, z.B. das Alter von *Melanie* in Relation 2 und b) „Widersprüche“, z.B. das Alter von *Christoph*.
Möchte man die Daten aus beiden Relationen in eine einzige Relation überführen, bzw. anfragen, stellt sich die Frage, wie man mit den auftretenden Konflikten umgeht. Dieses Problem wurde erstmals in [1] erwähnt. Seither gibt es verschiedene Lösungsansätze. Gleichwohl gibt es kein System und keine Technik, die ein Ergebnis wie in Abbildung 1 dargestellt erzeugt.

2 Data Merging in Integrationssystemen

In den meisten der bisher erstellten Systeme zur Integration von Daten wird die Problematik des Auftretens von Konflikten zwischen Informationsquellen nicht betrachtet. Einige, die dies dennoch tun, werden im Folgenden vorgestellt. Es gelingt jedoch keinem, sowohl Unsicherheiten zu beseitigen, als auch Widersprüche sinnvoll aufzulösen.
TSIMMIS: In TSIMMIS [4] werden semistrukturierte Daten aus verschiedenen Quellen zusammengeführt. Objektidentifikation findet dabei mittels semantischer ID's statt, die sich als Funktion mehrerer Attribute ergeben. Objekte mit gleicher semantischer ID werden zusammengeführt, fehlende Informationen ergänzt, Redundanzen entfernt und Widersprüche vermieden, indem pro Attribut eine bevorzugte Quelle festgelegt wird [11]. Dies geschieht im Mediator durch Angabe von Regeln in der Mediator-Spezifikationssprache von TSIMMIS.
Hermes: Auch in Hermes [5] wird die Integration von Daten im Mediator aus verschiedenen Quellen mittels Regeln bestimmt. Dazu werden 5 verschiedene Strategien genannt die Datenkonflikte aufzulösen: Die Entscheidung für das jüngere Datum, zwei verschiedene Arten der Entscheidung für ein Objekt anhand der Quelle aus der es stammt, die Entscheidung anhand des Wertes bei numerischen Daten (z.B. immer der kleinere Wert) und die Entscheidung für die „zuverlässigere“ Quelle. Hierbei bleibt allerdings eine genauere Definition von zuverlässig offen.
Data Cleansing Systeme: Bei Data Cleansing Systemen steht im Gegensatz zu Integrationssystemen die Bereinigung des Datenbestandes von Fehlern im Vordergrund. Der Integrationsaspekt, d.h. das Vorhandensein mehrerer Quellen und der Aspekt der Optimierung von Anfragen rückt in den Hintergrund.
Im System Potters' Wheel [12] interagieren Nutzer mit dem fehlerbehafteten Datenbestand, der ihnen als Tabelle dargestellt wird. Mittels Transformationen (z.B. Add, Drop, Merge, Split, ...) können die Daten solange verändert werden, bis sie keine Fehler mehr enthalten. Eine Komponente im System erkennt selbstständig Fehler in den Spalten und macht Nutzer darauf aufmerksam.
Im System Ajax [6] wird vom Nutzer ein Transformationsprozess definiert, den die Daten durchlaufen, um bereinigt zu werden. Der Prozess wird aus mehreren vorhandenen elementaren Transformationen zusammengesetzt. Die Bereinigung von Fehlern geschieht generell in drei Phasen, indem zuerst gleiche Objekte identifiziert (Matching), diese dann gruppiert (Clustering) und schliesslich zu einem Objekt zusammengeführt werden (Merging).
Kommerzielle RDBMS: Die meisten kommerziellen RDBMS (z.B. IBM DB2, Oracle) bieten die Möglichkeit zur Anbindung externer Quellen mittels Wrapper. Die Inhalte werden in Relationen importiert und können dann mittels Standard SQL-Statements weiterverarbeitet werden. Teil des neuen SQL2003 Standards [3] ist das Merge-Statement als abkürzende Schreibweise für eine Folge von Insert-/Update-/Delete-Statements. Damit ist es möglich, eine Tabelle um Datensätze zu erweitern, wobei in der Regel bestehende Datensätze (bestehender Schlüssel) geändert und neue Datensätze (neuer Schlüssel) hinzugefügt werden.

3 Data Merging Techniken

Neben den genannten Systemen gibt es auch eine Vielzahl von weiteren Techniken zur Integration von Daten. Diese sind Teil der relationalen Algebra oder Erweiterungen davon.

3.1 Relationale Operatoren

Die folgenden Techniken bauen auf relationaler Algebra als Grundlage auf, erweitern oder nutzen sie, um Daten aus unterschiedlichen Relationen zusammenzuführen. Tabelle 2 stellt die Operatoren anhand der nachfolgend genannten Kriterien einander gegenüber.

Informationserhaltend: Ein Operator ist informationserhaltend, wenn alle Informationen, die auch in den Ursprungsrelationen vorhanden sind, nach der Ausführung des Operators in der Ergebnisrelation erhalten bleiben.

Schlüsselerhaltend: Das Ergebnis eines schlüsselerhaltenden Operators enthält ein eindeutiges Schlüsselattribut in der Ergebnisrelation. Insbesondere erhält der Operator die Schlüsselwerte der Objekte aus den Ursprungsrelationen.

Umgang mit Konflikten: Es werden drei verschiedene Arten der Behandlung von Konflikten unterschieden: a) Konflikte werden *ignoriert*, d.h. es werden weder Unsicherheiten beseitigt, noch Widersprüche gelöst. b) Konflikte werden *vermieden*, d.h. Unsicherheiten werden beseitigt, Widersprüche aber nicht gelöst, sondern durch die (geschickte) Auswahl von Werten umgangen. c) Konflikte werden *gelöst*, d.h. Unsicherheiten werden beseitigt und Widersprüche sinnvoll aufgelöst.

Umsetzung: Einige der betrachteten Operatoren sind elementarer Bestandteil der relationalen Algebra, während andere mit ihrer Hilfe ausgedrückt werden können.

Union ($\cup$): Union vereinigt die Tupel zweier Union-kompatibler Relationen und entfernt exakte Duplikate [14]. Union ist für nicht Union-kompatible Relationen nicht definiert. Somit ist Union nicht schlüsselerhaltend, da nur Tupel entfernt werden, die in allen gemeinsamen Attributen übereinstimmen. Unsicherheiten und Widersprüche in den Daten werden ignoriert.
Outer Union ($\uplus$): Outer Union vereinigt nicht Union-kompatible Relationen, indem es die nicht gemeinsamen Attribute ergänzt und sie mit `Null`-Werten auffüllt [7]. Wie auch Union ist Outer Union nicht schlüsselerhaltend und Konflikte werden ignoriert. Outer Union ist kein eigener SQL Operator, er kann aber in SQL ausgedrückt werden.
Minimum Union ($\oplus$): Minimum Union wird von [7] definiert als Outer Union, dessen Ergebnis um subsumierte Tupel bereinigt wird. Ein Tupel t_1 subsumiert ein anderes Tupel t_2, wenn t_2 mehr $\perp$-Werte als t_1 enthält, ansonsten aber in allen Attributwerten mit t_1 übereinstimmt. Es gelten die selben Eigenschaften wie für Outer Union. Nur wenn alle Konflikte Unsicherheiten aufgrund subsumierter Tupel sind, ist der Operator schlüsselerhaltend. Widersprüche werden nicht aufgelöst. Minimum Union kann nicht mit Hilfe von SQL ausgedrückt werden, da die Entfernung subsumierter Tupel als Operator fehlt.
Natural Join ($\bowtie$): Der Natural Join fügt Tupel aus zwei Relationen zusammen, wenn alle gemeinsamen Attribute in ihren Werten übereinstimmen [14]. Selbst wenn gemeinsame Attribute gleich sind, gehen Informationen aus den Relationen verloren, da Objekte, die nicht in beiden Relationen enthalten sind, auch nicht Teil des Ergebnisses sind. Der Operator ist schlüsselerhaltend, Konflikte werden ignoriert.
Key Join ($\bowtie_{id=id}$): Im Unterschied zum Natural Join, fügt der Key Join Tupel aus zwei Relationen bereits dann zusammen, wenn das designierte Schlüsselattribut gleich ist (Theta-Equijoin auf dem Schlüsselattribut [14]). Trotzdem werden alle Attribute aus den Ursprungsrelationen übernommen. Daher ist der Operator sowohl informations-, als auch schlüsselerhaltend. Durch eine weitere geschickte Selektion von Attributen können Konflikte vermieden werden, ansonsten werden sie ignoriert.
Outer Join ($\overset{\circ}{\bowtie}$): Der Full Outer Join Operator erweitert den Natural Join um die Tupel die jeweils nur in einer der beiden Relationen vorhanden sind. Die fehlenden Attribute werden um $\perp$-Werte ergänzt [14]. Es existiert eine Variante, bei der ein Key Join statt eines Natural Join zugrunde gelegt wird, sowie eine Left und Right Variante, bei der nur um die Tupel aus der linken, bzw. rechten Relation erweitert wird. Der Operator ist nur in der Full Outer Join Variante mit Key Join informationserhaltend, aber in allen Varianten schlüsselerhaltend. Unsicherheiten werden beseitigt, Widersprüche bleiben jedoch bestehen.

Operator	*Informations-erhaltend*	*Schlüssel-erhaltend*	*Konflikte werden...*	*Umsetzung*
Union	ja, falls def.	nein, falls def.	ignoriert, falls def.	Standard-SQL
Outer Union	ja	nein	ignoriert	Übersetzung in SQL
Minimum Union	ja	nein/ja	ignoriert/vermieden	keine Übersetzung in SQL
Natural Join	nein	ja	ignoriert	Standard-SQL
Key Join	ja	ja	ignoriert	Standard-SQL
Outer Join	ja/nein	ja	vermieden	Standard-SQL
Match Join	ja/nein	nein/ja	vermieden	Übersetzung in SQL
Merge	ja	nein	vermieden	Übersetzung in SQL

Tabelle 2: Übersicht der Eigenschaften der relationalen Operatoren

Match Join: Dieser Operator wird im Rahmen des Systems AURORA definiert [16]. Er lässt sich als Outer Join über das Schlüsselattribut zwischen allen Wertekombinationen aller Attribute umschreiben. Die Attributwerte sind zur Durchführung des Joins jeweils um den Schlüsselwert ergänzt. Je nach Parametrisierung des Operators werden alle Informationen bzw. Schlüssel erhalten oder nicht. Unsicherheiten werden beseitigt, Widersprüche allerdings nicht aufgelöst.
Merge ($\boxtimes$), Prioritized Merge ($\lhd$): Auf dem Match Join Operator aufbauend wird in [8] der Merge Operator definiert. Er lässt sich als Vereinigung zweier Outer Joins in SQL umschreiben und zeigt, wie die SQL Funktion Coalesce zur Beseitigung von Unsicherheiten verwendet werden kann. Unsicherheiten werden beseitigt, aber Widersprüche nicht aufgelöst. Mit Hilfe der Coalesce Funktion kann auch eine Priorisierung der Quellen angegeben werden (Prioritized Merge).

3.2 Sonstiges

Ebenso wie die vorigen Operatoren basieren die folgenden Ansätze auf relationaler Algebra, führen allerdings keinen neuen Operator ein, sondern nutzen, erweitern, oder ändern die Semantik bestehender Elemente.
Benutzerdefinierte Aggregation: Bei diesem Ansatz werden Daten integriert, indem zuerst mittels ID's Gruppen gleicher Objekte gebildet werden, die dann durch Aggregation zu einem einzigen Objekt verschmolzen werden. In FraQL [13] werden auftretende Datenkonflikte durch die Verwendung von benutzerdefinierten Aggregationsfunktionen gelöst. Benutzerdefinierte Aggregation ähnelt damit stark der in Ajax verwendeten Vorgehensweise.
Da benutzerdefinierte Aggregation nicht Teil des SQL-Standards ist, stellt AXL [15] eine Möglichkeit dar, aufbauend auf SQL und Objektrelationalen DBMSen, benutzerdefinierte Aggregation z.B. zum Data Merging zu nutzen.
Zusätzliche Attribute: Zwei weitere Ansätze [2, 10] haben gemeinsam, dass der Relation zusätzliche Attribute zugefügt werden, mit Hilfe dessen die Zugehörigkeit zur integrierten Ergebnisrelation bestimmt wird. In einem Fall ist dies die Kennzeichnung durch ja/nein/vielleicht, in einem anderen die Angabe eines Wahrscheinlichkeitswertes. Die Semantik und auch die Implementierung der Standard-SQL-Operatoren wird erweitert, um mit diesen zusätzlichen Attributen umgehen zu können.
Coalesce: Eine weitere Möglichkeit zur Integration bietet die Verwendung der SQL Coalesce Funktion zusammen mit einem Join Operator. `coalesce(A,B,...,X)` liefert den ersten Wert zurück, der nicht `Null` ist. Werden als Parameter gleiche Attribute aus verschiedenen Relationen übergeben, beseitigt Coalesce Unsicherheiten, bzw. kann wie im Merge-Operator zur Priorisierung von Relationen eingesetzt werden. Widersprüche werden durch Coalesce nicht aufgelöst.

4 Zusammenfasung

Die Integration von Daten aus heterogenen Informationsquellen wird in unterschiedlichen Systemen auf verschiedene Art und Weise umgesetzt. Neben existierenden Systemen wurden Techniken vorgestellt, die auf relationaler Algebra basieren bzw. sie erweitern.
Ein Vergleich der verschiedenen Systeme und Techniken ergibt, dass die Anforderungen von Informations- und Schlüsselerhaltung, Konfliktlösung und relationaler Umsetzung von keinem System und keiner Technik gleichzeitig erfüllt werden.

Der Humboldt-Merger (HumMer) ist ein System zur Integration von Informationsquellen, das in der Arbeitsgruppe Informationsintegration der Humboldt-Universität zu Berlin entwickelt wird [9]. Anfragen in HumMer werden an mehrere heterogene Informationsquellen gestellt, dem Nutzer wird ein einzelnes integriertes Ergebnis präsentiert. Das System unterstützt mehrere Datenmodelle, insbesondere XML und relationale Informationsquellen.
Das Augenmerk bei HumMer liegt auf den Komponenten zur Objektidentifikation (OI), dem Zusammenfügen von Daten und Auflösen von Konflikten (DM) und der Optimierung von Anfragen, die OI- und DM-Operationen enthalten. Eine schematische Übersicht über HumMer zeigt Abbildung 1.

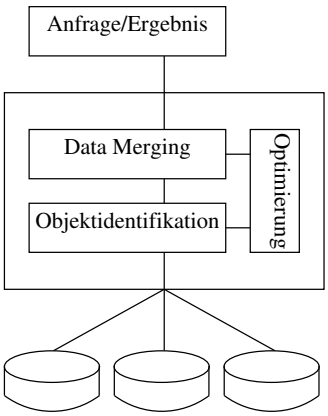

Abbildung 1: Schematische Darstellung von HumMer

Die in der Übersicht in Tabelle 2 dargestellten Operatoren mit ihren Eigenschaften dienen somit als Ausgangspunkt zur Definition des (relationalen) binären HumMer Data Merging Operators, der die gewünschten Eigenschaften informationserhaltend, schlüsselerhaltend und konfliktlösend besitzen soll. Desweiteren sollte er assoziativ sein. Ist der Operator assoziativ, spielt die Reihenfolge bei der Anwendung auf mehr als 2 Relationen keine Rolle.
Die Identifizierung von Objekten wird dabei in einem ersten Schritt der Vorverarbeitung vorrausgesetzt. Dies wird durch die OI-Komponente geleistet. Das gewünschte Ergebnis der Anwendung des Operators auf die Beispieldaten zeigt Tabelle 1. Darauf aufbauend kann die Optimierung von Anfragen, die sowohl Standard-SQL als auch Data Merging Operatoren enthalten, untersucht und verbessert werden. Dank an Felix Naumann für wertvolle Kommentare.

Literatur

[1] U. Dayal. Processing queries over generalization hierarchies in a multidatabase system. In *Proc. Int. Conf. on Very Large Data Bases (VLDB)*, pages 342–353, 1983.

[2] L. G. DeMichiel. Resolving database incompatibility: An approach to performing relational operations over mismatched domains. *IEEE Trans. Knowl. Data Eng.*, 1(4):485–493, 1989.

[3] A. Eisenberg et. al. Sql:2003 has been published. *SIGMOD Rec.*, 33(1):119–126, 2004.

[4] H. Garcia-Molina et. al. The tsimmis approach to mediation: Data models and languages. *J. Intell. Inf. Syst.*, 8(2):117–132, 1997.

[5] V. S. Subrahmanian et. al. Hermes: A heterogeneous reasoning and mediator system. Technical report, University of Maryland, 1995.

[6] H. Galhardas, D. Florescu, D. Shasha, and E. Simon. Ajax: An extensible data cleaning tool. In *Proc. of the Int. Conf. on Management of Data (SIGMOD)*, page 590, 2000.

[7] C. Galindo-Legaria. Outerjoins as disjunctions. In *Proc. of Int. Conf. on Management of Data (SIGMOD)*, pages 348–358, 1994.

[8] S. Greco, L. Pontieri, and E. Zumpano. Integrating and managing conflicting data. In *Revised Papers from the 4th Int. Andrei Ershov Memorial Conf. on Perspectives of System Informatics*, pages 349–362, 2001.

[9] http://www.informatik.hu-berlin.de/mac/hummer.

[10] E.-P. Lim, J. Srivastava, and S. Shekhar. Resolving attribute incompatibility in database integration: An evidential reasoning approach. In *Proc. of the Int. Conf. on Data Engineering (ICDE)*, pages 154–163, 1994.

[11] Y. Papakonstantinou, S. Abiteboul, and H. Garcia-Molina. Object fusion in mediator systems. In *Proc. of Int. Conf. on Very Large Data Bases (VLDB)*, pages 413–424, 1996.

[12] V. Raman and J. Hellerstein. Potter's wheel: An interactive data cleaning system. In *Proc. of Int. Conf. on Very Large Data Bases (VLDB)*, pages 381–390, 2001.

[13] K. Sattler, S. Conrad, and G. Saake. Adding Conflict Resolution Features to a Query Language for Database Federations. In *Proc. 3rd Int. Workshop on Engineering Federated Information Systems, EFIS*, pages 41–52, 2000.

[14] J. D. Ullman, H. Garcia-Molina, and J. Widom. *Database Systems: The Complete Book*. Prentice Hall PTR, 2001.

[15] H. Wang and C. Zaniolo. Using sql to build new aggregates and extenders for object- relational systems. In *Proc of Int. Conf. on Very Large Data Bases (VLDB)*, pages 166–175, 2000.

[16] L. L. Yan and M. Özsu. Conflict tolerant queries in aurora. In *Proc. of the 4th IECIS Int. Conf. on Cooperative Information Systems (CoopIS)*, page 279, 1999.