Data Preprocessing 1

Thema: Business Intelligence –
Teil I: OLAP & Data Warehousing

von Christian Merker

Betreuer: Dipl.-Inform. Michael Haustein

26. Juni 2003

Inhalt

1 Einleitung 3

2 Monitoring und Daten Extraktion 4

2.1 Realisierung 4

2.2 Techniken 5

2.2.1 Aktive Mechanismen 6
2.2.2 Replikationsmechanismen 6
2.2.3 Protokollbasierte Entdeckung 7
2.2.4 Anwendungsunterstütztes Monitoring 7

2.3 Extraktion der Daten 8

3 Datenintegration 10

3.1 Schwierigkeiten bei der Integration 10

3.2 Schemaintegration 10

3.2.1 Vorintegrationsphase 11
3.2.2 Vergleichsphase 12
3.2.3 Vereinheitlichungsphase 12
3.2.4 Restruktuierungsphase 13

3.3 Transformation der Daten 14

3.3.1 Schlüsselbehandlung 15
3.3.2 Vereinheitlichung von Zeichenketten 15
3.3.3 Vereinheitlichung von Datumswerten 17
3.3.4 Umrechnung von Maßeinheiten 17
3.3.5 Separierung / Kombination von Attributwerten 18
3.3.6 Berechnung abgeleiteter Werte 18

4 Data Cleaning 19

4.1 Behandlung von fehlerhaften Werten 19

4.2 Behandlung von Nullwerten 19

4.3 Redundanz 20

4.4 Duplikateliminierung (Sorted-Neighborhood-Methode) 21

5 Zusammenfassung 23

Referenzen 24

1. Einleitung

Große Firmen besitzen schon heute Terra- beziehungsweise Petabyte an Datenmengen, die in diversen Datenbanken gespeichert und für Analysezwecke verwendet werden. Durch die Globalisierung besitzen diese Firmen Zweigstellen in verschiedenen Ländern. Die in den Zweigstellen gewonnenen Daten werden in lokalen Datenbanken gespeichert und im Bedarfsfall von einem zentralen Data-Warehouse zur Analyse ausgelesen. Dabei können durch die Benutzung von verschiedenen Datenbanksystemen Probleme entstehen. Zum Beispiel kann sich die Repräsentation der Daten oder die Datentypen unterscheiden. Verfahren wie Datenintegration und Datenbereinigung (im Englischen Data Cleaning oder Data Cleansing genannt) beheben diese Probleme, indem sie die Daten während dem Auslesen in ein einheitliches Format transformieren. In der Datenintegrationsphase, auf die wir in Kapitel 3 ausführlich eingehen, werden die lokalen Schemadaten aus den Quellsystemen ausgelesen und auf ähnliche Strukturen hin untersucht. Anschließend wird aus diesen Informationen ein globales Schema für das Data-Warehouse erstellt.

Die Datenbereinigung untersucht die integrierten Daten auf Inkonsistenzen, inkorrekte Werte und Redundanzen und beseitigt diesen „Schmutz“ von den Daten. In Kapitel 4 werden Verfahren aufgezeigt, die zum Beispiel inkorrekte Werte oder Nullwerte mit Hilfe von Ähnlichkeitsfunktionen und anderen Verfahren beseitigen. Die Sorted-Neighborhood-Methode, die in Kapitel 4.4 vorgestellt wird, eignet sich besonders zum Aufspüren und Beseitigen von Redundanzen.

Eine ebenso wichtige Rolle in dem Prozess der Integration und Bereinigung von Daten spielen Monitore, die in Kapitel 2 vorgestellt werden. Ein Monitor ist dafür zuständig, die Änderungen von Daten in der Datenbank zu erkennen und alle für das Data-Warehouse interessanten Daten zu markieren. Dadurch müssen nur noch markierte Daten extrahiert werden und der Auslesevorgang wird beschleunigt.

Dass Datenintegration und Datenbereinigung heutzutage eine sehr wichtige Rolle in Data-Warehouse Systemen einnehmen, verdeutlichen die Schätzungen des Data-Warehouse Instituts, die davon ausgehen, dass amerikanischen Firmen 2001 durch „schmutzige“ Daten rund 600 Milliarden Dollar an Einnahmen entgangenen sind [Pete03]. Diese Zahl verdeutlicht, dass „schmutzige“ Daten weitaus mehr Schaden verursachen, als es auf ersten Blick erscheinen mag.

2. Monitoring und Daten Extraktion

Die Aufgabe von Monitoren ist es, die Daten in einem Datenbanksystem zu überwachen und Änderungen zu entdecken. Monitore werden üblicherweise in den Quellsystemen eingesetzt und stellen ein Bindeglied zwischen den Datenquellen und dem Data-Warehouse System dar. Der Monitor ist außerdem dafür verantwortlich, dass der Datenbeschaffungsprozess, dass heißt die Extraktion, die Transformation und der Ladevorgang der Daten angestoßen wird. Dabei werden natürlich nur die geänderten und für das Data-Warehouse relevanten Daten betrachtet.

2.1. Realisierung

Wie bereits angesprochen, haben Monitore die Aufgabe, die Änderungen auf relevanten Daten für das Data-Warehouse zu protokollieren und diese Informationen auf Anfrage des Warehouses weiterzuleiten. Lässt man die domänspezifischen Eigenschaften einmal außer Acht, so kann man zwei grundsätzliche Varianten für die Entdeckung von geänderten Daten unterscheiden:

- Die erste Möglichkeit besteht darin, dass der Monitor alle relevanten Änderungen in einer so genannten Delta-Datei speichert. Diese Datei ist ähnlich einer Log-Datei, wie sie beim Logging und Recovery vom Datenbanksystem verwendet wird [HäRa01]. Auf Anfrage vom Data-Warehouse wird diese Delta-Datei an das Warehouse übertragen und dort weiter analysiert (siehe Kapitel 3).
- Die andere Möglichkeit besteht darin, dass der Monitor sich bei einer Datenänderung nur als Hinweis merkt, dass die Datenquelle geändert wurde, jedoch nicht, welche Daten im Einzelnen von der Änderung betroffen sind. Dem Data-Warehouse werden diese Hinweise mitgeteilt, worauf das Data-Warehouse die Extraktionskomponente startet, die dann auf die Datenquelle zugreift, die Änderungen sucht und extrahiert.

Weiterhin sind die folgenden gegensätzlichen Aspekte bei der Gestaltung eines Monitors zu beachten:

- Nettoeffekt vs. Entdeckung aller Änderungen: Soll der Monitor das Data-Warehouse über alle Änderungen informieren, oder nur über den letzten aktuellen Zustand zur Ladezeit. Zum Beispiel würde das Einfügen eines Tupels und das anschließende Löschen dieses Tupels beim Nettoeffekt keinen Ein-

trag hinterlassen, während bei der anderen Methode zwei Einträge vermerkt werden würden.

- Benachrichtigung vs. Polling: Beim Polling muss der Monitor periodisch auf die Datenquelle zugreifen, um zu überprüfen, ob Änderungen entstanden sind. Wenn die Periodendauer sehr kurz ist, kann es durch die häufigen Zugriffe des Monitors zu Leistungseinbußen kommen, wird hingegen die Periodendauer zu groß, werden eventuelle Änderungen nicht erkannt (Beispiel: Einfügen eines Tupels und anschließendes Löschen desselben). Eine Benachrichtigung des Monitors durch die Datenquelle hat den Vorteil, dass jede Änderung erkannt wird und das System nicht unnötig belastet wird. Voraussetzung ist natürlich, dass die Datenquelle in der Lage ist, ihre Umgebung zum Beispiel mittels Triggern zu informieren.
- Internes vs. externes Monitoring: Im Fall des internen Monitoring stellt das Quellsystem hinreichende Monitoring Möglichkeiten zur Verfügung, die im Falle einer Datenänderung vom Data-Warehouse genutzt werden können. Dabei ist allerdings darauf zu achten, dass diese Funktionalitäten den Betrieb des Quellsystems nicht beeinträchtigen, beziehungsweise unnötig belasten. Beim externen Monitoring stellt das Quellsystem nur unzureichende oder keine Möglichkeiten im Falle einer Datenänderung zur Verfügung. Dann müssen Änderungen von außen entdeckt werden, zum Beispiel indem man die Daten mit periodisch erzeugten Extrakten vergleicht.

2.2. Techniken

Zur Realisierung von Monitoren stehen eine Reihe von verschiedensten Techniken zur Verfügung. Ihr Einsatz in einem Quellsystem hängt jedoch stark von den Charakteristika der einzelnen Datenquellen ab, das heißt auf welche Art und Weise die Informationen über eine Datenänderung zur Verfügung gestellt werden.
In [BaGü01] werden vier Techniken vorgestellt, die im Folgenden kurz erläutert werden sollen.

2.2.1. Aktive Mechanismen

Aktive Mechanismen können vorher definierte Situationen erkennen und für die Situation festgelegte Aktionen ausführen. Als Grundkonzept sind die so genannten ECA-Regeln (Event, Condition, Action) zu nennen. Tritt ein definiertes Ereignis ein, so wird die Bedingung evaluiert und im Falle der Erfülltheit eine Aktion ausgeführt. Abbildung 2.1 zeigt ein Beispiel für ECA-Regeln.

```
EVENT          Kontoeinzahlung
CONDITION      Kontostand > 5000 €
ACTION         Kontozinsen auf 3 % erhöhen
```

Abbildung 2. 1

Bei einer Einzahlung auf ein Konto tritt das Ereignis Kontoeinzahlung ein, anschließend wird überprüft, ob der Kontostand eine bestimmte Schwelle (5000 €) überschreitet, wenn ja, dann wird eine Aktion (Zinsen auf 3 % erhöhen) ausgeführt.
Die Anwendungsprogramme können durch den Einsatz von ECA-Regeln wesentlich vereinfacht werden, da das Datenbankmanagementsystem (DBMS) für die Ausführung der Regeln zuständig ist. Nachteil ist allerdings, dass nicht alle Datenbanksysteme diesen eleganten Mechanismus zur Verfügung stellen und eine intensive Verwendung von Triggern zu Leistungseinbußen im Quellsystem führt.

2.2.2. Replikationsmechanismen

Bei Replikationsmechanismen werden alle relevanten Datenänderungen in eine spezielle Datenbank repliziert, welche zur Ladezeit des Data-Warehouse abgefragt wird. Die folgenden zwei Realisierungsmöglichkeiten sollen einen Einblick in die Idee der Replikationsmechanismen geben:

- Snapshot: Ein Snapshot ist eine lokale Kopie von Daten aus einer oder mehreren Tabellen und wird, ähnlich wie eine Sicht, mittels SQL-Anfrage definiert. Auf Snapshot-Daten kann nur ein lesender Zugriff erfolgen. Eine Aktualisierung der Daten kann entweder vollständig, das heißt die Anfrage wird erneut ausgeführt, oder inkrementell erfolgen, zum Beispiel mit einer Snapshot-Log Datei. Um einen Monitor basierend auf der Snapshot Technik zu implementieren, müssen auf allen relevanten Tabellen des Quellsystems Snapshots sowie die zugehörigen Snapshot-Logs definiert werden. Die Snapshot-Logs, die Informationen über die Änderungen seit dem letzten

Snapshot enthalten, werden zum Beispiel in Oracle8i dazu benutzt, um eine Delta-Datei zu erstellen [BaGü01].

- Datenreplikation: Analog zur Snapshot-Technik werden bei der Datenreplikation die replizierten Daten in eine (Ziel-)Tabelle mittels SQL-Anfragen geschrieben und verwaltet. Bei der Realisierung eines Monitors ist vor allem die inkrementelle Aktualisierung dieser Zieltabellen von Bedeutung, das heißt der Monitor überwacht die Datenbank hinsichtlich Änderungen und schreibt diese Änderungen in so genannte Delta-Tabellen, wobei auch Änderungen von Transaktionen vermerkt werden, die später abbrechen oder ein Rollback durchführen. Die Änderungsinformationen entnimmt der Monitor den Logging und Recovery Daten der Datenbank. Anschließend überprüft der Monitor die Delta-Tabellen und löscht alle Einträge von Transaktionen die nicht korrekt abgeschlossen wurden. Die restlichen Einträge werden in konsistenten Delta-Tabellen gespeichert. Um das Quellsystem nicht allzu sehr zu belasten, können die Delta-Tabellen auf andere Rechner ausgelagert werden [BaGü01].

2.2.3. Protokollbasierte Entdeckung

Protokollbasierte Entdeckungsmechanismen nutzen die Protokolldatei, die für das Datenbankrecovery erstellt wird, um Änderungen an den Datenbeständen zu entdecken. Dies ist allerdings nur möglich, wenn die Log-Datei ausreichend Informationen zur Verfügung stellt und von außen zugänglich ist. Bei einem Aufruf eines Data-Warehouses wird dann die Log-Datei übertragen und anschließend im Data-Warehouse untersucht. Die Datenquelle wird zwar dadurch nicht belastet, jedoch stellen der Datentransport und die Untersuchung der Daten einen erheblichen Aufwand dar. Die protokollbasierte Endeckung setzt genaue Kenntnisse über die Struktur der Log-Datei voraus, was durch den system- und versionsabhängigen Aufbau der Log-Datei erschwert wird und somit ein weiterer Nachteil ist.

2.2.4. Anwendungsunterstütztes Monitoring

Besteht keine Möglichkeit den Monitor durch eine der oben genannten Techniken zu implementieren, so bleibt oftmals nur noch die Möglichkeit, das Monitoring durch die Anwendungsprogramme zu realisieren. Die Programme müssen so geändert beziehungsweise erweitert werden, dass bei Änderungen von Daten durch das Programm

zusätzlich eine Delta-Datei erstellt wird und im Bedarfsfall diese an das Data-Warehouse übertragen wird. Problematisch sind jedoch meistens alte Programme, die aus heutiger Sicht schlecht entworfen und dokumentiert sind und durch neue Programme ersetzt werden müssen. Die neuen Programme verwenden vor allem Verfahren mit Zeitstempeln, das heißt zu den Daten werden zusätzlich Felder mit Zeitinformationen gespeichert, die durch den Monitor zur Ladezeit überprüft werden und bei einem Zeitstempel, der größer dem Zeitpunkt des letzten Ladevorgangs ist, markiert werden.

2.3. Extraktion der Daten

In der Extraktionsphase werden alle relevanten Daten, die geändert wurden, aus den verschiedenen Datenquellen in das Data-Warehouse übertragen. Dabei spielt die eingesetzte Monitor Technik (siehe Kapitel 2.2) eine entscheidende Rolle, da sie die Qualität der Informationen über relevante Änderungen identifiziert. Die Zeitpunkte, zu denen eine Extraktion durchgeführt wird, sind abhängig vom Einsatzgebiet und können je nach Bedarf unterschiedlich gewählt werden. Man unterscheidet folgende prinzipiellen Vorgehensweisen [BaGü01]:

- Periodisch: Die Extraktion geschieht in bestimmten Zeitintervallen, wobei die Länge des Zeitintervalls von der Dynamik der Daten abhängt. Ein Beispiel für ein kurzes Aktualisierungsintervall sind die Börsenkurse bei Nachrichtensendern, bei denen die Daten alle 15 Minuten aktualisiert werden. Im Falle von Preis- oder Produktänderungen kann ein größeres Zeitintervall zur Aktualisierung der Daten angesetzt werden, beispielsweise monatlich oder halbjährlich.
- Anfragegesteuert: In diesem Fall wird die Extraktion durch eine explizite Anfrage von außen angestoßen, zum Beispiel kann durch den Aufruf des Administrators oder durch eine Anfrage des Data-Warehouse nach geänderten Daten die Extraktion angestoßen werden.
- Ereignisgesteuert: Die Extraktion wird durch ein bestimmtes Ereignis angestoßen, zum Beispiel könnte das Erreichen einer bestimmten Anzahl von Datenänderungen solch ein Ereignis sein. Streng genommen sind auch die beiden ersten Vorgehensweisen ereignisgesteuert, das erste beginnt nach Eintreten eines bestimmten Zeitereignisses, das zweite wird durch ein externes Ereignis gestartet.

- Sofort: Manche Systeme stellen sehr hohe Anforderungen an die Aktualität der Daten, zum Beispiel bei Börsenkursen kann es notwendig sein, dass eine Änderung im Quellsystem sofort auch im Data-Warehouse zur Verfügung stehen soll. Die Daten im Data-Warehouse sind damit immer genauso aktuell wie die in den Quellsystemen.

Die technische Realisierung der Extraktion hängt sehr stark von der Softwareumgebung und der verwendeten Hardware in den Quellsystemen und im Data-Warehouse ab. Es gibt jedoch standardisierte Schnittstellen wie ODBC, die verwendet werden, um die Realisierung zu vereinfachen.

3. Datenintegration

In der Datenintegrationsphase werden die Daten wie auch die Schemata aus den Quellsystemen im Data-Warehouse auf die Qualitätsanforderungen des Anwenders angepasst. Dieses Kapitel beschäftigt sich mit der Integration der lokalen Schemata der Quellsysteme zu einem globalen Schema im Data-Warehouse. Danach wird ein kurzer Überblick darüber gegeben, welche Transformationen bezüglich der Daten beachtet werden müssen.

3.1. Schwierigkeiten bei der Integration

Die grundsätzlichen Probleme bei der Integration der Daten in das Data-Warehouse entstehen auf der einen Seite durch unterschiedliche Datenmodelle und Datenbanksysteme, auf der anderen Seite treten Schemakonflikte auf, die beispielsweise durch unterschiedliche Interpretationen der Miniwelt während der Realisierung der Datenbank entstanden sind. Ebenfalls wird durch unterschiedliche Bezeichnungen von Entities die Integration von Daten aus verschiedenen Quellen erschwert. Oftmals wird dadurch ein offensichtlicher Zusammenhang zwischen Entity-Typen nicht oder nur schwer erkannt. Eine solche Situation erfordert dann einen manuellen Eingriff, um falsche Zuordnungen zu vermeiden.

3.2. Schemaintegration

Die Schemaintegration beschäftigt sich mit der Vereinigung der lokalen Schemata der einzelnen Quellsysteme zu einem globalen Schema des Data-Warehouse. Dabei werden Konflikte zwischen den einzelnen Schemata entdeckt und nach Möglichkeit beseitigt. Diese Aufgabe kann noch immer nicht vollständig automatisiert werden und erfordert daher noch immer manuelle Eingriffe. Glücklicherweise muss eine Schemaintegration nur beim ersten Laden der Daten in das Data-Warehouse oder bei einer Schemaevolution durchgeführt werden [Tesc99]. In [BaLN86] wird die Schemaintegration in 4 Phasen gegliedert:

- Vorintegrationsphase
- Vergleichsphase
- Vereinheitlichungsphase
- Restrukturierungsphase

Die Schemaintegration ist ein sehr umfassendes und sehr gut erforschtes Themengebiet, deshalb soll an dieser Stelle mit Hilfe eines einfachen Beispiels aus

[MaBR01] ein Eindruck darüber vermittelt werden, was in den einzelnen Phasen geschieht. Tiefere Einblicke in dieses Themengebiet erhält man in [BaLN86].

3.2.1. Vorintegrationsphase

In der Vorintegrationsphase werden die Schemata auf den einzelnen Quellsystemen analysiert und anschließend ausgewählt, welche Schemateile integriert werden sollen. Darüber hinaus werden weitere Informationen für die Integration gesammelt und eine generelle Integrationsstrategie festgelegt [JLVV03].
In unserem Beispiel gehen wir von den Schemata in Abbildung 3.1 aus:

```
Table PO = (POShipTo(City, Street); POBillTO(City,
Street); POLines(Count, Item(Line, Qty, UoM))

Table PurchaseOrder(Items(ItemCount, Item(ItemNumber,
Quantity, UnitOfMeasure); DeliverTo(Address(Street,
City)); InvoiceTo(Address(Street, City)))
```

Abbildung 3. 1

Die Schemata wirken auf den ersten Blick sehr unübersichtlich, durch die Analyse kann zum Beispiel eine Baumstruktur, wie in Abbildung 3.2 darstellt, erzeugt werden.

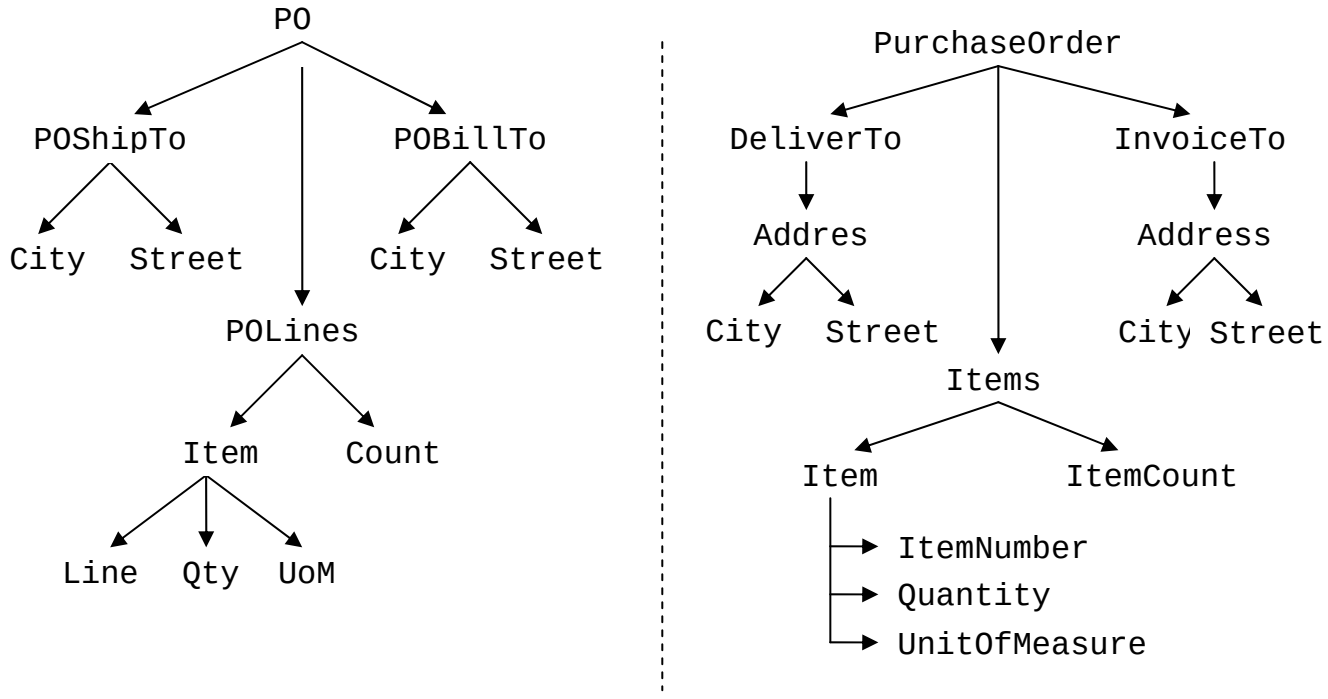

Abbildung 3. 2 [MaBR01]

3.2.2. Vergleichsphase

In dieser Phase wird untersucht, ob es zwischen den Schemata Korrespondenzen gibt. Dies geschieht mit Hilfe von Thesauri und Wörterbüchern. Weiterhin wird nach möglichen Konflikten zwischen den Schemata gesucht [JLVV03]. In [BaLN86] werden 4 Hauptfehlerklassen unterschieden:

- Namenskonflikte: Sie entstehen, wenn in verschiedenen Schemata unterschiedliche Bezeichnungen für dieselben Daten verwendet werden. Dadurch entstehen Homonym- und Synonymfehler (siehe Kapitel 3.3.1).
- Semantische Konflikte: Diese Konflikte treten auf, wenn verschiedene Abstraktionsebenen in den Quellsystemen für die Daten verwendet werden.
- Strukturkonflikte: Sie entstehen, wenn bei der Entwicklung der Datenbank unterschiedliche Strukturen zur Abbildung der realen Welt benutzt wurden.
- Datenmodellkonflikte: In den letzten 15 Jahren wurden Datenbanken fast ausschließlich auf Basis von relationalen Datenbankmodellen entworfen. Trotzdem gibt es immer noch Altsysteme, die auf hierarchischen oder netzwerkartigen Datenmodellen basieren und daher andere Strukturen wie die relationalen Datenbanken besitzen, was zu Konflikten führen kann.

In unserem Beispiel würde diese Phase der Schemaintegration eine Reihe von Ähnlichkeiten mit Hilfe Thesauri und fachspezifischen Abkürzungslisten feststellen (Abbildung 3.3).

```
1. Qty     →   Quantity
2. UoM     →   UnitOfMeasure
3. Bill    →   Invoice
```

Abbildung 3. 3

3.2.3. Vereinheitlichungsphase

Das Ziel dieser Phase ist es, die lokalen Schemata für die anschließende Zusammenführung zu modifizieren und in der Vergleichsphase entdeckte Konflikte zu beseitigen. Bei der Eliminierung der Konflikte wird oftmals der Einsatz einer Person mit Wissen aus dem Anwendungsgebiet benötigt. Dies ist natürlich zeitaufwendig, doch wie bereits erwähnt, muss diese Prozedur nur beim erstmaligen Laden der Daten und bei Änderungen in den Schemata durchgeführt werden [JLVV03].

Im Beispiel können wir nun durch die in der Vergleichsphase gewonnenen Informationen (Abbildung 3.3) leicht unsere Schemata vereinheitlichen (Abbildung 3.4).

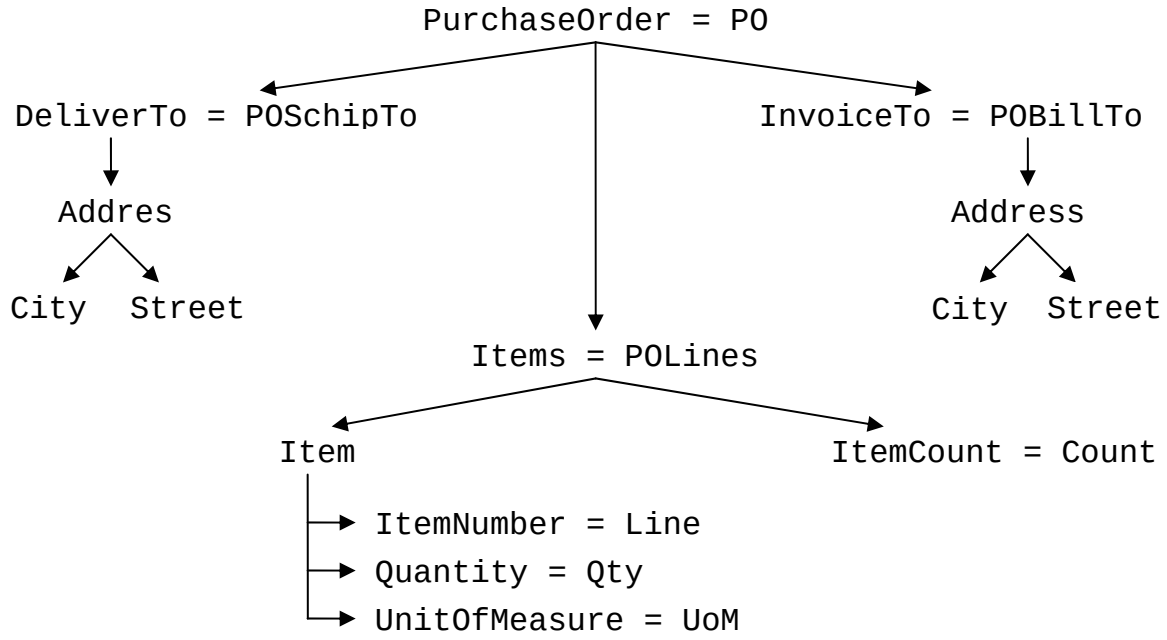

Abbildung 3. 4

Es ist auch leicht ersichtlich, dass das Attribut `Line` dem Attribut `ItemNumber` entsprechen muss, da in beiden Strukturen der Vaterknoten identisch ist und die anderen beiden Attribute sich nach Abbildung 3.3 entsprechen. Weiterhin wird schnell klar, dass die Attribute `City` und `Street` unter `POBillTo` im zweiten Schema auf `City` und `Street` unter `InvoiceTo` abgebildet werden müssen, da `Invoice` ein Synonym für `Bill` ist.

3.2.4. Restruktuierungsphase

In der letzten Phase der Schemaintegration wird ein globales Schema erstellt, das im Data-Warehouse zum Einsatz kommt. Nach der Erstellung des globalen Schemas wird dieses auf seine Qualität bezüglich Vollständigkeit (completeness), Minimalität (minimality), Verständlichkeit (understandability) und Korrektheit (correctness) untersucht [SaCo00]. Im Folgenden stellen wir kurze Erläuterung zu den vier Begriffen dar:

- Vollständigkeit: Alle Konzepte aus den lokalen Schemata müssen im integrierten Schema vorhanden sein. Es dürfen keine Konzepte aus lokalen Schemata verloren gegangen sein.

- Minimalität: Existieren Konzepte beziehungsweise Daten, die in mehreren lokalen Schemata modelliert sind und sich auf dasselbe Objekt der realen Welt beziehen, dürfen im integrierten Schema nur einmal auftreten.
- Verständlichkeit: Das integrierte Schema sollte verständlich sein und aussagekräftige Bezeichnungen benutzen.
- Korrektheit: Alle im integrierten Schema vorhandenen Informationen müssen mindestens in einem lokalen Schema semantisch äquivalent vorhanden sein. Es sind nur konsistente Ergänzungen der bestehenden Schemata erlaubt, das heißt sind während der Integration neue Beziehungen zwischen Schemata hinzugekommen, so dürfen diese nicht im Widerspruch zu Informationen aus den lokalen Schemata stehen.

In Abbildung 3.5 ist das fertige, integrierte Schema zu unserem Beispiel zu sehen.

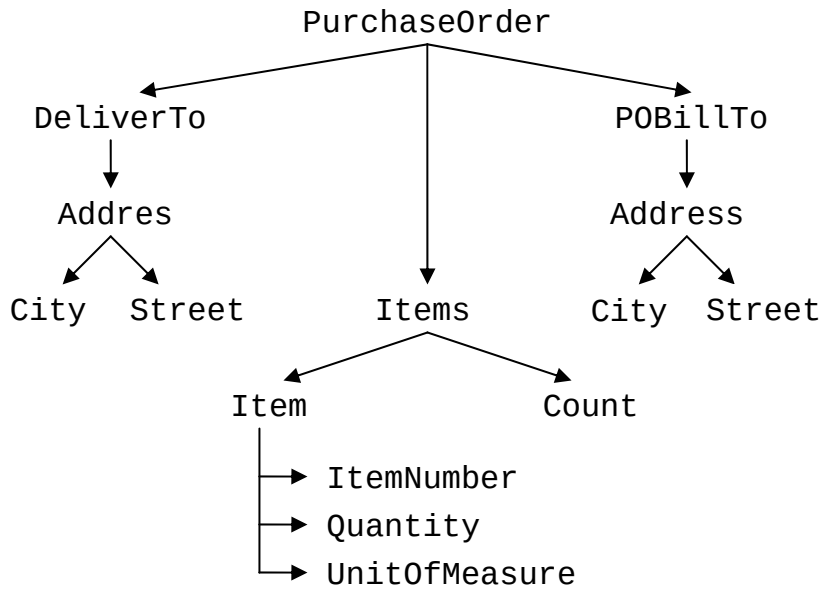

Abbildung 3. 5

3.3. Transformation der Daten

Dieses Kapitel beschäftigt sich mit der Integration der eigentlichen Daten in das Data-Warehouse. Die Daten werden dafür in eine vorher vom Anwender festgelegte Form transformiert.

3.3.1. Schlüsselbehandlung

Die Primärschlüssel aus den Schemata sind zwar bezogen auf das jeweilige Schema eindeutig, jedoch kann es bei der Zusammenführung von Daten aus verschiedenen Quellen zu Überschneidungen kommen. Es ist dann notwendig, ein zusätzliches Attribut in das Schema einzufügen, um eine eindeutige Identifizierung zu ermöglichen. Heutige Systeme, wie beispielsweise Oracle 8i besitzen SQL-Kommandos wie `create sequence` die beim Aufruf automatisch einen eindeutigen Wert zurückliefern.

Die fehlende Eindeutigkeit von Schlüsseln ist aber oft nicht das einzige Problem. Hinzu kommt oftmals, dass Schlüssel eine implizite Semantik besitzen. Der Schlüssel könnte aus einer Produktbezeichnung und dem Einführungsjahr bestehen (zum Beispiel Prod97). Bei einer Übernahme in das Data-Warehouse dürfen solche Informationen natürlich nicht verloren gehen. Wenn die implizite Semantik bekannt ist, ist es möglich, bei der Transformation diesen Wert zu extrahieren und in mehrere Attribute zu speichern. Eine weitere Schwierigkeit entsteht bei der Entscheidung, ob zwei Attribute aus verschiedenen Quellsystemen sich auf dieselbe Entität in der realen Welt beziehen. Dieses Problem erfordert meistens gute Kenntnisse des Anwendungsgebiets und einen manuellen Eingriff, um die Gefahr von Synonym- und Homonymfehler zu reduzieren. Synonymfehler entstehen, wenn mehrere Bezeichnungen für ein Objekt existieren (zum Beispiel Telefon und Fernsprecher). Homonymfehler treten auf, wenn eine Bezeichnung für verschiedene Objekte benutzt wird (zum Beispiel Bank als Geldinstitut oder als Gartenbank). Um solche Fehler heute zu einem gewissen Grad automatisch zu erkennen, benutzt man statistische Verfahren, die einen Ähnlichkeitswert zwischen den Datensätzen berechnen. Überschreitet die Ähnlichkeit einen gewissen Schwellenwert, so werden die Datensätze einander zugeordnet.

3.3.2. Vereinheitlichung von Zeichenketten

Die Vereinheitlichung von Zeichenketten wird durchgeführt, um unter anderem Synonym- und Homonymfehler wie sie in 3.3.1. beschrieben wurden zu verringern. Die Zeichen werden dabei beispielsweise alle in Groß- oder Kleinbuchstaben transformiert, oder sprachspezifische Sonderzeichen wie Umlaute werden durch mehrbuchstabige Zeichenketten ersetzt. Abbildung 3.6 zeigt Beispiele dafür, in welcher Weise Zeichenketten vereinheitlicht werden können.

Küchengerät	→	Kuechengeraet
Videorekorder	→	VIDEOREKORDER
VCR	→	Videorecorder

Abbildung 3. 6

Auch die Eliminierung von überflüssigen Leerzeichen und Tabulatoren werden hierbei beachtet. Bei der Ersetzung von Abkürzungen ist darauf zu achten, dass ein unternehmensweit einheitliches Vokabular zum Beispiel von Fachbegriffen verwendet wird. Es ist natürlich auch möglich, vorausgesetzt man besitzt das nötige Anwendungswissen, weitere Transformationen durchzuführen, beispielsweise könnte ein Attribut „Name“, der den Familiennamen und Vornamen einer Person beinhaltet, in zwei separate Attribute „Nachname“, „Vorname“ aufgeteilt werden.
Die Transformation sorgt für eine einheitliche Basis für Vergleiche der Merkmalsausprägungen, wodurch die Gefahr von Synonymfehler verringert wird. Jedoch können dadurch auch Merkmalsausprägungen verloren gehen und Homonymfehler entstehen.
Um Schreib- und Hörfehler zu kompensieren, ist es sinnvoll so genannte phonetische Codes zu bilden (Abbildung 3.7) und einen Vergleich der Datensätze auf Basis dieser Codes durchzuführen.

(nach Kölner Phonetik, Variante von 1975)

Videorekorder	→	Fedeurekurder
Videorecorder	→	Fedeurekurder

Abbildung 3. 7

Ein ebenfalls interessantes Verfahren, um ähnlich klingende Wörter zu vergleichen, ist der von Magaret Odell und Robert Russel entwickelte SoundEx-Algorithmus. Der Grundgedanke bei diesem Verfahren ist die Annahme, dass Wörter die ähnlich klingen auch semantisch ähnlich sind. Der Algorithmus erzeugt aus einer Zeichenkette einen dreistelligen Code, indem er jedem Buchstaben eine vorher festgelegte Codezahl zuweist. Sind nach der Erzeugung die Codes zweier Wörter gleich, so wertet der Algorithmus diese als ähnlich aus. Das Verfahren soll an dieser Stelle nicht weiter vertieft werden, eine ausführliche Beschreibung findet sich in [NARA00]

3.3.3. Vereinheitlichung von Datumswerten

Bei Datumswerten kann man bei heutigen Datenbanksystemen zwischen der internen und externen Darstellung unterscheiden. Die interne Darstellung ist statisch während die externe Darstellung den jeweiligen Anwenderbedürfnissen angepasst werden kann, beispielsweise um ein spezielles landesspezifisches Format darzustellen. Besitzt das Quellsystem diese Unterscheidung in interne und externe Darstellung, so ist keine Transformation notwendig, da die externe Darstellung entsprechend dem Data-Warehouse angepasst werden kann. Ist dies nicht der Fall und das Data-Warehouse erfordert eine spezielles Datumsformat (zum Beispiel „TT.MM.JJJJ“), so muss der Datumswert konvertiert werden. In Altsystemen wurde das Datum meist als Zeichenkette gespeichert, in diesem Fall muss eine Typkonvertierung des Datums vorgenommen werden.

Des Weiteren kann es vorkommen, dass die Quell- oder Zieldatenbank das Datum in einzelnen numerischen Angaben für Tag, Monat und Jahr speichert. In diesem Fall müssen die Einzelwerte kombiniert beziehungsweise separiert werden (zum Beispiel „TT“, „MM“, „JJJJ“ → „MM-TT-JJJJ“). Größere Probleme bereiten Datumsangaben, bei denen nur Teile des Datums gespeichert wurden, zum Beispiel in Altsystemen nur zwei Stellen für die Jahreszahl, oder nur die Angabe von Monat und Jahr. Im ersten Fall müssen die Jahreszahlen auf vier Stellen erweitert werden, wobei das richtige Jahrhundert zu beachten ist (normalerweise19xx oder 20xx). Im zweiten Fall fügt man für die fehlenden Werte einfach Dummies ein, etwa immer den 1. oder den 15. des Monats.

3.3.4. Umrechnung von Maßeinheiten

Die Umrechnung von Maßeinheiten betrifft meistens numerische Werte, da diese oftmals eine Maßeinheit wie Länge oder Gewicht besitzen. Mit Hilfe von Umrechungstabellen ist die Umrechnung, wie die Beispiele in Abbildung 3.8 zeigen, eine triviale Aufgabe.

```
10 inch   →   25,4 cm   →   0,254 m
10 km     →   10.000m
30 mph    →   48,279 km/h
```

Abbildung 3. 8

Allerdings ist die Transformation nur trivial, wenn die Einträge auch einen Vermerk über die verwendete Maßeinheit besitzen. Ist dies nicht der Fall, so wird von der Integration dieser Daten abgeraten.
Verwenden Quell- und Zielsystem beide die gleiche Maßeinheit, zum Beispiel metrische Längenangaben, so kann es trotzdem notwendig sein die Skalierung der Maßeinheit vorzunehmen (zum Beispiel 25,4 cm → 0,254m).

3.3.5. Separierung / Kombination von Attributwerten

Manchmal kann es, wie bereits bei der Vereinheitlichung von Datumswerten, vorkommen, dass in den Quellsystemen zusammengefasste Attributwerte im Data-Warehouse in einzelne Attribute aufgespaltet werden müssen. Diesen Vorgang nennt man Separierung während der umgekehrte Fall, also das Zusammenfügen von einzelnen Attributen aus den Quellen als Kombination von Attributwerten bezeichnet wird. Beispiele hierfür können die Datumszerlegung aus Kapitel 3.3.3. oder die Zerlegung von Schlüsselwerten mit impliziter Semantik sein (siehe Kapitel 3.3.1.). Die Zerlegung beziehungsweise Kombination von Attributen kann nach einer vorher festgelegten Regel oder Berechnungsvorschrift geschehen.

3.3.6. Berechnung abgeleiteter Werte

Unter Umständen kann es sinnvoll sein (um die Zugriffszeit von bestimmten Anfragen zu verringern), aus den Attributwerten der Quelldatenbanken neue Attribute abzuleiten. Angenommen in einer Relation gibt es 1.000.000 Tupel, und für die Datenanalyse wird häufig der Mittelwert über diese Tupel benötigt. Dann ist es sinnvoll, den Mittelwert einmal beim Laden der Daten zu bestimmen und zu speichern und bei anschließenden Anfragen auf den gespeicherten Wert zuzugreifen. Dadurch werden die Anfragen um ein vielfaches effizienter. Man sollte aber weiterhin auch die Werte, aus denen die neuen Werte abgeleitet wurden mitspeichern, da sie eventuell für andere Auswertungen noch gebraucht werden können. Die Ableitung von neuen Werten kann dabei weitaus komplexer und schwieriger sein, als das abschließende Beispiel in Abbildung 3.9 vermuten lässt:

```
Preis incl MWST = Preis ohne MWST * 1,16
```

Abbildung 3. 9

4. Data Cleaning

In der Bereinigungsphase werden die inkorrekten, unvollständigen und inkonsistenten Daten entdeckt und nach Möglichkeit behandelt. Ziel ist es sicherzustellen, dass im Data-Warehouse eine bereinigte Datenbank vorzufinden ist und die Qualität der Daten erhöht wird.

4.1. Behandlung von fehlerhaften Werten

Unter fehlerhaften Werten versteht man beim Data Cleaning Datenwerte, die unvollständige oder falsche Angaben (zum Beispiel bei Adressen) beinhalten. Null-Werte gehören ebenfalls in diesen Bereich, ihre Behandlung wird im Kapitel 4.2. ausführlich erläutert.

Bei unvollständigen und fehlerhaften Daten kann mit Hilfe von Daten aus anderen Datenquellen ein Abgleich durchgeführt und somit die Daten ergänzt beziehungsweise korrigiert werden. Des Weiteren kann man auch domänenspezifische Wörterbücher einsetzen, beispielsweise im Pharmaziebereich, wenn Medikamente oder Wirkstoffe betrachtet werden. Solche Änderungen, die sich auf ein abgegrenztes Anwendungsgebiet beziehen, können heute durch Programme vollautomatisch durchgeführt werden, eine domänenunabhängige Validierung erfordert jedoch meist wieder zeitaufwendige, manuelle Eingriffe.

4.2. Behandlung von Nullwerten

Nullwerte können durch unterschiedliche Situationen entstanden sein [BaGü01], zum Beispiel:

- In der realen Welt gibt es keinen Wert für das Attribut. Ein Beispiel wäre die Haltbarkeitsdauer von nicht verderblichen Produkten.
- Der Wert des Attributs existiert zwar in der realen Welt, jedoch ist er zum Zeitpunkt der Eingabe nicht bekannt oder wurde aus bestimmten Gründen nicht erfasst. Ein Beispiel wäre die Nicht-Erfassung von optionalen Kundendaten in den Quellsystemen, die aber bei Analysen im Data-Warehouse benötigt werden (zum Beispiel Kundenalter).

Zur Behandlung von Nullwerten stehen eine Reihe von Möglichkeiten zur Verfügung. Die in [HaKa01] angesprochenen Verfahren werden im Folgenden kurz vorgestellt:

- Tupel ignorieren: Diese Methode ist sicherlich nicht die Effektivste, da bereits Tupel, in denen nur ein Attribut unbekannt ist, ignoriert werden.

Dadurch gehen viele wichtige Werte für die Analyse der Daten verloren. Allerdings ist diese Methode am einfachsten zu realisieren.

- Fehlende Werte manuell einfügen: Diese Methode ist sehr zeitaufwendig und erfordert den Einsatz von Personen mit Kenntnissen aus dem Anwendungsgebiet. Sie ist außerdem in großen Datenbanken mit mehreren Millionen Einträgen und vielen Nullwerten nicht mehr realisierbar.
- Globale Konstanten verwenden: Bei dieser Methode wird allen unbekannten Werten ein konstanter Wert (zum Beispiel „Unbekannt“) zugewiesen. Ein Problem entsteht bei der Analyse, wenn diese konstanten Werte nicht herausgefiltert werden. Dann würde das Data-Mining Programm missverständlicherweise diesen Wert als wichtig annehmen, da er sehr häufig auftritt. Auch diese Methode ist einfach zu realisieren, findet allerdings in der Realität nur geringen Einsatz.
- Mittelwert: Diese Methode weist jedem Nullwert den Mittelwert über diese Spalte in der Datenbank zu. Dadurch werden Statistiken nicht verfälscht. Der Einsatz dieser Methode macht jedoch nur auf numerischen Attributen wirklich Sinn. Den Mittelwert über eine Adressspalte durchzuführen ergibt nicht unbedingt eine sinnvolle Adresse, dieses Problem kann man mit der nächsten Methode besser lösen.
- Häufigster Wert: Bei diesem Verfahren wird dem unbekannten Attribut der Wert zugeordnet, der für dieses Attribut am häufigsten auftritt. Auch diese Methode verändert Statistiken nur geringfügig und beim Einsatz für das oben genannte Adressproblem liefert es zwar nicht die richtige, aber zumindest eine sinnvolle Adresse. Durch diese Vorteile findet dieses Verfahren auch die häufigste Anwendung in heutigen Data-Warehouse Systemen.

4.3. Redundanz

Eine Redundanz liegt vor, wenn sich die Daten in einem Tupel des gleichen oder eines anderen Datensatzes wiederholen. Redundanz entsteht häufig durch eine fehlende Normalisierung des realisierten Schemas. Jedoch bedeutet das Vorhandensein von Redundanz nicht, dass die Daten von schlechter Qualität sind. Sie kann nur zu Problemen führen, wenn durch die Speicherung dieser redundanten Daten in verschiedenen Datensätzen oder Datenquellen die Konsistenz der Daten nicht mehr sichergestellt ist. Um dieses Problem zu umgehen, kann man Verfahren einsetzen,

die Duplikate finden und eliminieren. In Kapitel 4.4. wird ein einfaches Verfahren zur Duplikateliminierung vorgestellt.

4.4. Duplikateliminierung (Sorted-Neighborhood-Methode)

Die Sorted-Neighborhood-Methode benutzt die folgenden drei Schritte, um Duplikate zu entdecken und zu eliminieren:

- Schlüssel generieren: In diesem Schritt wird die Datenbank durchsucht und zu jedem Tupel ein Schlüssel generiert. Der Schlüssel entsteht, indem aus jedem Attribut des Tupels eine bestimmte Anzahl an Zeichen herausgenommen wird und diese zu dem Schlüssel konkateniert werden. Es erweist sich als sinnvoll [RaSa99], dass man die ersten drei Konsonanten aus Zeichenketten und die ersten drei Ziffern von Zahlenwerten nimmt. Den generierten Schlüssel fügt man anschließend als ein weiteres Attribut dem Tupel hinzu. Wie man in Abbildung 4.1 erkennt, ähnelt der erzeugte Schlüssel einer Art Hashsumme über das Tupel. Die Idee hinter dieser Generierung liegt darin, dass ähnliche Tupel den gleichen beziehungsweise einen sehr ähnlichen Schlüssel erhalten, die Schlüsselgenerierung ist demnach eine Ähnlichkeitsfunktion. Besitzen zwei Tupel gleiche beziehungsweise sehr ähnliche Schlüssel, so ist die Wahrscheinlichkeit groß, dass es sich um redundante Daten handelt.

ID	Name	Vorname	Adresse	Key
4711	Schmidt	Fritz	Hauptstr. 168	471SCHFRTHPT
0815	Maier	Kurt	Auf dem Acker 4	081MERKRTFDM
4711	Schmid	Fritzchen	Hauptstrasse 168	471SCHFRTHPT
0815	Mayer	Kurt W.	Auf dem Acker 4	081MYRKRTFDM

Abbildung 4. 1

- Sortieren: In diesem Schritt wird der Datenbestand nach dem generierten Schlüssel sortiert. Dies hat den Vorteil, dass man im Anschluss eine Gruppierung nach ähnlichen Tupeln hat und nicht immer die ganzen Daten nach Ähnlichkeit durchsuchen muss. Beim verwendeten Sortieralgorithmus ist darauf zu achten, dass ein Algorithmus mit geringer Laufzeitkomplexität,

beispielsweise Quicksort oder Mergesort, eingesetzt wird. Nach dem Sortieren ähnelt unsere Tabelle der Abbildung 4.2.

ID	Name	Vorname	Adresse	Key
4711	Schmidt	Fritz	Hauptstr. 168	471SCHFRTHPT
4711	Schmid	Fritzchen	Hauptstrasse 168	471SCHFRTHPT
0815	Maier	Kurt	Auf dem Acker 4	081MERKRTFDM
0815	Maier	Kurt	Auf dem Acker 4	081MERKRTFDM

Abbildung 4. 2

- Mischen: Im letzten Schritt der Sorted-Neighborhood-Methode werden die ähnlichen Daten nun zusammengemischt. Betrachtet man Abbildung 4.2 so stellt sich die Frage, welches Tupel ausgewählt werden soll. Man hat an dieser Stelle zwei Möglichkeiten, entweder man führt einen manuellen Eingriff durch, oder man legt fest, dass zum Beispiel das erste Tupel jeder Gruppe auswählt wird. Bei der Ähnlichkeitsüberprüfung können auch die anderen Attribute des Tupels mit einbezogen werden, um das Verfahren zu verbessern. Beispielsweise kann man, wie in Abbildung 4.3 verdeutlicht, eine Reihe von Bedingungen festlegen, mit deren Erfülltheit zwei Tupel als ähnlich angesehen werden.

```
IF T1.Schlüssel = T2. Schlüssel
AND T1.name gleich T2.name
AND die Vornamen sich nur gering unterscheiden
AND die Adressen eine Ähnlichkeit > 90% haben
```

Abbildung 4. 3

Anmerkung: Die Ähnlichkeit zwischen Zeichenketten kann man auch durch die Anzahl der unterschiedlichen Zeichen, die phonetische Gleichheit, das heißt klingen sie gleich, oder durch den Buchstabenabstand auf der Tastatur (zum Beispiel Dmith = Smith, da „D“ und „S“ auf einer QWERTZ Tastatur nebeneinander liegen) festlegen.

5. Zusammenfassung

Nachdem alle geänderten und relevanten Daten durch den Monitor entdeckt und markiert worden sind, werden die Daten in der anschließenden Extraktionsphase aus den Quellsystemen in das Data-Warehouse übertragen.

Die darauf folgende Schema- und Datenintegrationsphase sorgt dafür, dass alle Schemata aus den Quellsystemen in vier Schritten (Vorintegrationsphase, Vergleichsphase, Vereinheitlichungs- und Restruktuierungsphase) zu einem globalen Schema im Data-Warehouse zusammengefasst werden. Alle extrahierten Daten werden auf eine einheitliche Form gebracht, beispielsweise indem alle Zeichenketten auf Großbuchstaben und die Datumswerte in ein einheitliches Format konvertiert werden.

Die Datenbereinigung befreit die Daten im Data-Warehouse von Inkonsistenzen, inkorrekten Werten und Redundanzen. Zum Beispiel werden Nullwerte durch den Wert ersetzt, der für das Attribut am häufigsten auftritt, oder es wird eine globale Konstante verwendet. Zur Eliminierung von Duplikaten haben wir den Sorted-Neightborhood Algorithmus kennen gelernt. Dieser Algorithmus erstellt einen Schlüssel aus den Attributeinträgen eines Tupels um damit einen Ähnlichkeitsvergleich zwischen zwei oder mehreren Tupeln durchzuführen. Wenn ein bestimmter Schwellenwert überschritten wird, so erkennt der Algorithmus, dass es sich bei den ähnlichen Tupeln um das gleiche Objekt in der Realität handelt. Redundante Tupel werden anschließend aus der Datenbank entfernt.

Nach erfolgreichem Abschluss der Datenintegration und der Datenbereinigung, können die nun „sauberen“ Daten im Data-Warehouse für Auswertungen verwendet werden.

Referenzen

BaGü01 Bauer, Andreas; Günzel, Holger
Data-Warehouse Systeme; 1. Auflage; 2001

BaLN86 Batini, C.; Lenzerini, M.; Navathe S.B.
A comparative analysis of Methodologies for database schema integration. 1986

HaKa01 Hai, J.; Kamber, M.;
Data Mining: Concepts and Techniques; 2001

HäRa01 Härder, Theo; Rahm, Erhard
Datenbanksysteme – Konzepte und Techniken der Implementierung; 2. Auflage; 2001

JLVV03 Jarke, Matthias; Lenzerini Maurizio; Vassilion, Yannis; Vassiliadis, Panos;
Fundamentals of data warehouses; 2. Edition 2003

MaBR01 Madhavan, Jayant; Berstein, Philip A.; Rahm, Erhard
Generic Schema Matching with Cupid;
Proceedings of the 27th VLDB Conference, 2001
http://dol.uni-leipzig.de/pub/2001-28

NARA00 National Archives and Records Administration NARA
The Soundex Indexing System; 2000

Pete03 Peterson, Tommy
Data Scrubbing by the Numbers; 2003
http://www.computerworld.com/databasetopics/data/story/0,10801,78260,00.html

RaSa99 Raisinghani, Vijay T.; Sarawagi, Sunita
Cleaning Methods in Data-Warehousing; 1999
http://www.it.iitb.ac.in/~rvijay/seminar/dwhclean.ps.gz

SaCo00 Sattler, Kai-Uwe; Conrad, Stefan
Vorlesung Data-Warehouse-Technologien; 2000
http://wwwiti.cs.uni-magdeburg.de/iti_db/lehre/dw/

Tesc99 Teschke, Michael;
Datenbankkonsistenz in Data-Warehouse Systemen; Dissertation 1999