

Humboldt-Universität zu Berlin

Institut für Informatik



Studienarbeit

ManDup

Ein Werkzeug zur manuellen Duplikaterkennung

Daniel Mauter und Erik Witzmann

Betreuer: Dipl.-Ing. (BA) Melanie Weiß

Inhaltsverzeichnis

1	Vorstellung der Aufgabe	4
1.1	Einführung	4
1.2	Anforderung	4
2	Benutzung	6
2.1	Installation der Webanwendung ManDup	6
2.1.1	Vorraussetzungen	6
2.1.2	Installationsdateien	6
2.1.3	Installation	6
2.2	Hinweise zur Webanwendung ManDup für Benutzer	7
2.2.1	An-/Abmelden	7
2.2.2	Aufgaben bearbeiten	8
2.2.2.1	Auswahl der Aufgabe	8
2.2.2.2	Bearbeitung	8
2.2.3	Einstellungen	8
2.3	Hinweise zur Webanwendung ManDup für Administratoren	9
2.3.1	Filter	9
2.3.1.1	Was ist ein Filter	9
2.3.1.2	Implementieren eines neuen Filters	9
2.3.1.3	Einen neuen Filter bei ManDup anmelden	10
2.3.2	Aufgabenverwaltung	10
2.3.2.1	Anlegen einer neuen Aufgabe	10
2.3.2.2	Aufgabe neu initialisieren	11

2.3.2.3	Löschen einer Aufgabe	11
2.3.3	Benutzerverwaltung	11
2.4	Hinweise zum Verwenden der Bearbeitungsergebnisse	11
2.4.1	Die Relation WORK_RESULTS	11
3	Spezielle Konzepte	13
3.1	Verwaltung der Bearbeitungsreihenfolge für jeden Nutzer	14
3.2	Erkennen und Verwalten der Transitivität	15
3.2.1	Transitivität bei zwei Relationen	15
3.2.2	Transitivität bei einer Relation	15
3.3	Speicherung der Daten	17
4	Architektur	19
4.1	Verwendete Technologie	19
4.2	Komponenten der Anwendung	19
4.2.1	Die MVC-Komponenten	20
4.2.2	Die Work-Komponente	20
4.2.2.1	Verwendete Klassen und deren Funktionalität	22
4.2.2.2	Zusammenspiel	22
5	Ausblick	25

Kapitel 1

Vorstellung der Aufgabe

1.1 Einführung

Bei der Duplikaterkennung setzt man sich das Ziel, Datensätze zu identifizieren, die das gleiche Real-Welt Objekt beschreiben. Das Problem ist, dass im Allgemeinen auch ungleiche Datensätze das gleiche Objekt beschreiben können. Ursachen hierfür sind unter anderem unterschiedliche Schreibweisen, Fehler in mindestens einem Datensatz usw. . In der Arbeitsgruppe Informationsintegration werden Methoden zur automatischen Duplikaterkennung erforscht. Zur Verifikation benötigt man Datensätze, bei denen die Duplikate schon bekannt sind. Um dies möglichst komfortabel und fehlerfrei zu erreichen, soll ein Tool entwickelt werden. Die Anforderungen an dieses Tool sind zu einem Teil in einer Aufgabenstellung formuliert gewesen, zum Anderen auch erst aus Gesprächen mit der Arbeitsgruppe entwickelt worden.

1.2 Anforderung

Die Applikation soll Web-basiert sein, so dass es mehreren Benutzern gleichzeitig möglich ist, mit einem Browser an dem Suchen von Duplikaten zu arbeiten. Die Duplikaterkennung soll am Beispiel einer relationalen Filmdatenbank geschehen. Dabei soll die Möglichkeit bestehen, eine Relation auf Duplikate zu untersuchen oder zwei jeweils duplikatfreie Relationen untereinander zu vergleichen. Sollen zwei verschiedene Relationen verglichen werden, muss die Möglichkeit bestehen, ein Matching vorzugeben, das die Attribute der Relationen in Beziehung setzt.

Um die Anzahl der zu vergleichenden Tupel zu verringern, muss ein Filter implementiert werden, der Tupelpaare, die einem vorgegebenen Maß genügen, automatisch als unmögliches Duplikat identifiziert. Das Maß dieses Filters soll später ohne Aufwand ausgetauscht werden können. Es besteht in diesem Bereich die

Forderung nach einer generischen Implementierung. Des weiteren soll die Datenbankverbindung zu den Relationen, das Datenbankschema sowie die Relationen selber austauschbar sein.

Jedes Tupelpaar muss von zwei verschiedenen Bearbeitern beurteilt werden. Dabei sollen diese zwischen den Kategorien Duplikat, mögliches Duplikat und kein Duplikat wählen. Im Falle eines Duplikates muss der Duplikatgrund kategorisiert werden. Die Duplikatkategorien sollen erweiterbar sein. Um weitere Vergleiche zu sparen soll die Transitivität der Duplikatrelation benutzt werden. Konflikte die zwischen den Bearbeitern entstanden sind, sollen durch einen als Spezialist ausgezeichneten Bearbeiter aufgelöst werden.

Die Ergebnisse sollen abgespeichert werden, so dass ein zeitlich willkürliches Arbeiten sowie Statistiken über den so entstandenen Daten möglich werden. Eine effiziente Implementierung bezüglich der internen Repräsentation der Daten ist angestrebt.

Annahmen an die zu durchsuchenden Relationen

1. Es existiert ein eindeutiges Attribut vom Typ Integer.
2. Zu vergleichende Tupel, die neu in die Datenbank eingefügt werden, erhalten einen höheren Wert beim eindeutigen Integerattribut.
3. Werden zwei verschiedene Relationen verglichen, ist jede für sich genommen duplikatfrei.

Kapitel 2

Benutzung

2.1 Installation der Webanwendung ManDup

2.1.1 Voraussetzungen

- Apache Tomcat Server Version 5.0.28
- IBM DB2 Version 8.1 (und entsprechender JDBC Treiber)

2.1.2 Installationsdateien

mandup.ddl Diese noch anzupassende Datei enthält ein SQL-Script und beschreibt das von der Anwendung verwendete Datenbankschema und den benötigten initialen Datenbankbestand

mandup.war Archiv der Webanwendung

2.1.3 Installation

1. Erstellen Sie eine Datenbank, die von Mandup benutzt werden darf.
2. Legen Sie einen Benutzer an mit dem die Webanwendung sich zur Datenbank verbinden kann.
3. Ersetzen Sie in dem Script mandup.ddl alle Vorkommen von <DBNAME> mit dem Namen der Datenbank.
4. Ersetzen Sie in dem Script mandup.ddl alle Vorkommen von <DBUSER> mit dem Benutzernamen.

5. Führen Sie das angepasste Script `mandup.ddl` in einem DB2 Befehlsinterpreter aus.
6. Kopieren Sie wenn nötig den JDBC Treiber von DB2V8.1 `db2jcc.jar` in das Verzeichnis `%TOMCAT_HOME%/common/lib`
7. Spielen Sie das Archiv `mandup.war` auf den Server.
8. Konfigurieren Sie den verwendeten Verbindungspool in der Datei `%MANDUP_HOME%/META-INF/context.xml`. Insbesondere achten Sie bitte darauf, dass die richtige Verbindungs-URL zur Datenbank, der richtige Benutzer sowie das richtige Passwort angegeben sind.
9. Starten Sie die Webanwendung.

2.2 Hinweise zur Webanwendung ManDup für Benutzer

Die Webanwendung ManDup wurde für die manuelle Duplikaterkennung implementiert.

Ein Administrator der Anwendung kann verschiedene Aufgaben anlegen, welche dann von Benutzern bearbeitet werden sollen. Eine Aufgabe ist das Identifizieren von Duplikaten in einer Relation oder das Identifizieren von Duplikaten zwischen zwei bereits duplikatfreien Relationen.

In diesem Handbuch soll die Bedienung der Webanwendung ManDup für normale Benutzer erklärt werden. Informationen zur Administration der Anwendung entnehmen Sie bitte dem Abschnitt *Hinweise zur Webanwendung Mandup für Administratoren*.

2.2.1 An-/Abmelden

- Um die Webanwendung benutzen zu können müssen Sie zunächst einen Account bei dem Administrator beantragen.
- Mit einem gültigen Account können Sie sich dann auf der Hauptseite anmelden.
- Sollten Sie nach der Anmeldung längere Zeit inaktiv sein, werden Sie automatisch abgemeldet.
- Melden Sie sich bitte immer ab, bevor Sie die Seite verlassen.

2.2.2 Aufgaben bearbeiten

2.2.2.1 Auswahl der Aufgabe

Nach der Wahl des Menüpunktes *Aufgabe bearbeiten* müssen Sie zunächst eine Aufgabe wählen, die Sie bearbeiten möchten. Dabei gibt es zwei mögliche Situationen für jede Aufgabe in der Liste:

normale Bearbeitung Sie sind ein normaler Bearbeiter der Aufgabe.

Konflikte lösen Sie wurden als Spezialist dieser Aufgabe ausgewählt und müssen bei der Bearbeitung eventuell entstandene Konflikte lösen.

Zu jeder Aufgabe gibt es einen Knopf *Status* der Auskunft über den Bearbeitungsstand(bis auf Konflikte) zur jeweiligen Aufgabe erteilt.

2.2.2.2 Bearbeitung

Angekommen auf der eigentlichen Bearbeitungsseite, finden Sie oben eine Tabelle bestehend aus drei Zeilen. In der ersten sind die Attributnamen dargestellt, in der zweiten die entsprechenden Attributwerte des ersten Tupels und in der dritten die des zweiten Tupels. Sie sollen nun beurteilen ob das angegebene Tupelpaar ein Duplikat ist oder nicht. Wählen Sie *ist kein Duplikat*, wenn Sie meinen, dass es sich bei dem Tupelpaar um kein Duplikat handelt. Wenn Sie meinen ein Duplikat entdeckt zu haben, wählen Sie *ist Duplikat* und zusätzlich eine passende Duplikatkategorie. Sollten Sie keine passende Kategorie für das vorliegende Duplikat vorfinden, so wählen Sie *neue Kategorie hinzufügen* und geben in das benachbarte Textfeld die neue Kategorie ein.

Klicken Sie auf *Weiter* um das nächste Tupelpaar zu beurteilen. Sollten Sie die Bearbeitung der Aufgabe beenden wollen, so klicken Sie bitte auf *Speichern und Beenden*.

Für den Fall, dass Sie sich einmal bei der Beurteilung vertan haben, können Sie mit dem Knopf *Zurück* das zuletzt bearbeitete Tupel erneut beurteilen.

2.2.3 Einstellungen

Hier haben Sie die Möglichkeit Ihr persönliches Paßwort zu ändern. Bitte bedenken Sie bei der Wahl des Paßworts, dass dieses beim Anmeldevorgang **unverschlüsselt** übertragen wird. Wählen Sie deshalb auf keinen Fall ein bereits für andere Accounts verwendetes.

2.3 Hinweise zur Webanwendung ManDup für Administratoren

Im Folgenden werden Hinweise zum Benutzen von ManDup gegeben, die nur den Administrator betreffen. Diesem stehen unter dem Menüpunkt Einstellungen weitere Möglichkeiten zur Verfügung, welche in diesem Abschnitt vorgestellt werden.

2.3.1 Filter

2.3.1.1 Was ist ein Filter

Ein Filter schaut auf jedes potentielle Duplikat und entscheidet, ob es manuell beurteilt werden muss oder nicht. Dadurch kann eine große Menge von Tupelpaaren ignoriert werden, die sicher kein Duplikat sind.

2.3.1.2 Implementieren eines neuen Filters

Will man eine neue Relation bzw. zwei neue Relationen nach Duplikaten durchsuchen, so muß zuerst ein Filter für diese Relation/en implementiert werden. Dazu muß die Vaterklasse *Filter* abgeleitet werden.

Vorlagen für Filter finden Sie in dem Verzeichnis
`/home/studienarbeiten/mandup/WEB-INF/classes/mandup/filters`.

Verwenden Sie die Datei `Filter4OneRelation.java` als Vorlage für die Duplikatsuche in einer Relation und `Filter4TwoRelations.java` für die Duplikatsuche zwischen zwei **sauberen** Relationen. Die Vorlagen sind dokumentiert. Im Wesentlichen kommt es darauf an, die Parameter für den Datenbankzugriff anzugeben, sowie eine abstrakte Methode aus der Vaterklasse zu implementieren, welche für zwei Tupel entscheidet, ob sie Duplikate sind oder nicht.

Legen Sie den neuen Filter in das oben genannte Verzeichnis, kompilieren Sie die Klasse und setzen Sie die Zugriffsrechte für die .class-Datei so, dass der Tomcat-Server auf die Datei zugreifen kann.

Also:

1. In das Verzeichnis gehen:
`cd /home/studienarbeiten/mandup/WEB-INF/classes/mandup/filters`
2. Kopieren der entsprechenden Vorlage:
z.B. `cp Filter4OneRelation.java MyFilter.java`
3. Editieren der Klasse `MyFilter.java`:

- (a) Umbenennen des Konstruktors
 - (b) Setzen der Parameter
 - (c) Implementieren der Methode *maybeDuplicate(..)*
4. Kompilieren der Klasse `MyFilter`
- ```
javac -classpath /home/studienarbeiten/mandup/WEB-INF/classes/
MyFilter.java
```
5. Setzen der Zugriffsrechte:
- ```
chmod 755 MyFilter.class
```

2.3.1.3 Einen neuen Filter bei ManDup anmelden

Nachdem Sie wie oben beschrieben einen neuen Filter implementiert haben, können Sie diesen bei der Webanwendung für die Benutzung freigeben. Melden Sie sich dazu als Administrator ein und gehen zu *Einstellungen/Filter hinzufügen*.

Dort wählen Sie einen Namen für den Filter, geben den Klassennamen, die Anzahl der Relationen, sowie eine Beschreibung des Filters an. Wenn Sie dann auf *Filter hinzufügen* klicken, wird der Filter nach erfolgreichem Bestehen einiger Tests in den Datenbankbestand aufgenommen und kann für das Anlegen von Aufgaben verwendet werden.

2.3.2 Aufgabenverwaltung

2.3.2.1 Anlegen einer neuen Aufgabe

Legt man eine neue Aufgabe an muss man folgende Punkte abarbeiten:

1. Eintragen eines Namen für die Aufgabe, mit dem sie später identifiziert werden kann.
2. Auswahl eines Filters. Dieser muss für eine Aufgabe implementiert und bei ManDup angemeldet worden sein.
3. Auswahl eines Aufgabentyps. Dabei gibt es momentan die Möglichkeit Transitivität zu deaktivieren oder zu aktivieren.
4. Angabe eines Ausgabeschemas für die Tupelpaare. Hier wird festgelegt wie einem Benutzer später Tupelpaare zum Vergleich angeboten werden.
5. Aus allen bisher eingegebenen Duplikatgründen können beliebig viele für die neue Aufgabe übernommen werden.
6. Aus der Benutzerliste muss ein Spezialist ausgewählt werden.

Hat man sich durch diese Punkte durchgearbeitet, erscheint eine Übersichtsseite, die alle Angaben zusammenfasst. Nach dem Bestätigen, wird die Aufgabe angelegt. Je nach Filter und Menge der zu filternden Tupelpaare kann das anlegen sehr lange dauern. Man sollte sich daher über die Komplexität des Filteralgorithmus Gedanken machen, wenn sehr viele Tupel verglichen werden sollen.

2.3.2.2 Aufgabe neu initialisieren

Wurden in die Datenbank mit den zu vergleichenden Tupel neue Elemente eingefügt, können diese über eine neue Initialisierung berücksichtigt werden. Um dies zu veranlassen muss man nur unter *Einstellungen/ Aufgabe neu initialisieren* die Aufgabe auswählen und bestätigen. Der Filter untersucht nur neue Tupelkombinationen und fügt ggf. manuell zu vergleichende Paare in die Datenbank ein. Dieser Prozess kann nur fehlerfrei laufen, wenn für jedes zu filternde Tupel eine fortlaufende Id angelegt wurde, d.h. ein neu eingefügtes Tupel hat die größte Id.

2.3.2.3 Löschen einer Aufgabe

Beim Löschen einer Aufgabe, muss man zunächst die Aufgabe aus der Liste aller Aufgaben auswählen und dann auf den Knopf *Löschen* drücken. Es werden alle Einträge die speziell diese Aufgabe betreffen aus der Datenbank gelöscht.

2.3.3 Benutzerverwaltung

Unter dem Menüpunkt *Einstellungen* besteht die Möglichkeit Nutzer hinzuzufügen. Es wird somit eine Benutzerliste geführt, deren Mitglieder sich an der Bearbeitung sämtlicher Aufgaben beteiligen können.

2.4 Hinweise zum Verwenden der Bearbeitungsergebnisse

In diesem Abschnitt wird beschrieben, wo man die gesammelten Beurteilungen zu einer Aufgabe findet und wie diese zu interpretieren sind.

2.4.1 Die Relation WORK_RESULTS

In dieser Relation werden sämtliche Beurteilungen gespeichert. Die nun folgende Tabelle listet alle Attribute der Relation auf.

Attributname	Bedeutung
TASK_ID	Fremdschlüssel zur Aufgabe
FIRST_TUPLEID	ID des ersten Tupels
SECOND_TUPLEID	ID des zweiten Tupels
FIRST_EDITOR	ID des 1. Bearbeiters
FIRST_RESULT	Beurteilung des 1. Bearbeiters
FIRST_DUPLICATEREASON	Duplikatkategorie des 1. Bearbeiters
SECOND_EDITOR	ID des 2. Bearbeiters
SECOND_RESULT	Beurteilung des 2. Bearbeiters
SECOND_DUPLICATEREASON	Duplikatkategorie des 2. Bearbeiters
SPECIAL_RESULT	Beurteilung des Spezialisten
SPECIAL_DUPLICATEREASON	Duplikatkategorie des Spezialisten
INITNUMBER	Laufnummer der Initialisierung

Alle Attribute besitzen den Wertebereich Integer.

Zu jedem beurteilten Tupelpaar einer Aufgabe existiert genau ein Tupel in der Relation. Die Attribute TASK_ID, FIRST_TUPLEID und SECOND_TUPLEID stellen zusammen also einen Schlüsselkandidaten dar.

Ein Tupel kann in den Spalten FIRST_RESULT, SECOND_RESULT und SPECIAL_RESULT die Werte 0, 1 und 2 annehmen. Die Werte haben folgende Bedeutung:

- 0 Das Tupelpaar ist ein Duplikat.
- 1 Das Tupelpaar ist kein Duplikat.
- 2 Das Tupelpaar ist eventuell ein Duplikat.

Wurde ein Tupelpaar von einem Bearbeiter als Duplikat eingestuft, so findet man in den Attributen FIRST_DUPLICATEREASON, SECOND_DUPLICATEREASON und SPECIAL_DUPLICATEREASON einen Verweis zur angegebenen Duplikatkategorie.

Die Werte des Tupels in den Attributen SPECIAL_RESULT und SPECIAL_DUPLICATEREASON sind nur verschieden von *null*, wenn ein Konflikt zwischen den Beurteilungen vorlag und deshalb der Spezialist der Aufgabe das Tupelpaar bearbeiten mußte.

Kapitel 3

Spezielle Konzepte

In diesem Kapitel soll gezeigt werden, mit welchen Konzepten probiert wurde, die Anforderungen zu erfüllen. Eine Schwierigkeit lag darin zu gewährleisten, dass jedes Tupel zweimal betrachtet wird, jedoch immer von zwei verschiedenen Benutzern. Zudem schien es angemessen, die Tupel in einer für die Bearbeitung vorteilhaften Reihenfolge an die Benutzer weiterzugeben. Dem Bestreben nach Effizienz wurde in diesem Zusammenhang versucht, durch eine möglichst schnelle Berechnung des nächsten zu bearbeitenden Tupels, gerecht zu werden. Der Lebenslauf eines Tupels kann demnach wie folgt beschrieben werden:

Zunächst befindet es sich in der Ursprungsdatenbank, deren Verbindung bei der Konfiguration einer neuen Aufgabe hinterlegt wurde. Bei der Aufgabenkonfiguration wurde ebenfalls ein Filter festgelegt, welcher Tupelpaare wegfiltert, die sicher kein Duplikat sind.¹ Alle Tupelpaare, die den Filter passieren, werden in der Arbeitsdatenbank von ManDup gespeichert. Auf der Grundlage der Extension der Arbeitsdatenbank, wird beim Initialisieren eine Indexstruktur aufgebaut, die es ermöglicht, größtenteils ohne weitere Datenbankabfrage zu entscheiden, welches Tupelpaar jeder Benutzer als nächstes zu bearbeiten hat. Die Indexstruktur erspart ebenfalls das Kennzeichnen von Tupeln in der Datenbank, die an Benutzer weitergereicht wurden. Bearbeitet ein Benutzer das an ihn gereichte Tupel, wird sein Ergebnis in der Arbeitsdatenbank gespeichert und er bekommt ein weiteres Paar. Sollte die Sitzung eines Benutzers abbrechen, ohne dass sein letztes Tupel bearbeitet wurde, gelangt dieses in eine spezielle Liste, wodurch es für andere Benutzer frei gegeben wird, so sie es nicht schon selber bearbeit

¹Ursprünglich sollte immer ein Tupelpaar angefordert und für den Fall, dass es nicht gefiltert wird, dem Benutzer zur Bearbeitung vorgelegt werden und ansonsten ein neues Tupelpaar angefordert werden. Es stellte sich jedoch heraus, dass durch die Komplexität des vorgegeben Filteralgorithmuses, der Arbeitsfluss entschieden beeinträchtigt wurde. Somit wurde das komplette Filtern der Bearbeitung voran gestellt. Hierbei wird in Kauf genommen, dass vom Zeitpunkt des Anlegens einer Aufgabe bis zur Möglichkeit die Bearbeitung zu beginnen, viel Zeit vergehen kann. Dies hängt direkt mit der Kardinalität der zu vergleichenden Relationen und der Komplexität des gewünschten Filteralgorithmus zusammen.

haben. Eine weitere Liste ermöglicht es, einem Nutzer bei Fehleingabe einen Bearbeitungsschritt zurück zu gehen. Dabei wird sein aktuelles Bearbeitungstupel für ihn reserviert.

3.1 Verwaltung der Bearbeitungsreihenfolge für jeden Nutzer

Jedes Tupelpaar soll zweimal beurteilt werden, jedoch darf ein Benutzer ein Tupel maximal einmal bewerten.

Bei der Bearbeitung aller Tupel, ist es nicht erwünscht das Kreuzprodukt zu speichern. Daher werden die Tupelpaare aus einer inneren und einer äußeren Relation zusammengesetzt. Wird eine Relation mit sich selber verglichen, ist sie sowohl Innere als auch Äußere. Ist die innere Relation ungleich der Äußeren, wird das erste Tupel der äußeren Ergebnismenge festgehalten und mit allen Tupeln der inneren Ergebnismenge verglichen. Ist man bei der inneren Relation am Ende nimmt man das nächste Tupel der äußeren Ergebnismenge und vergleicht es wieder mit allen Tupel der inneren Ergebnismenge. Man ist am Ende, wenn man das letzte Tupel der Inneren mit dem letzten Tupel der äußeren Relation verglichen hat.

Ist die innere Relationen gleich der Äußeren, wird jedes Tupel der äußeren Ergebnismenge mit jedem dahinterstehenden Tupel der gleichen Ergebnismenge verglichen. Man ist am Ende, wenn das vorletzte Tupel mit dem Letzten verglichen wurde.

Da alle Tupelpaare zweimal beurteilt werden sollen, gibt es jedes Paar von Äußerer und innerer Ergebnismenge auch zweimal. Alle Benutzer arbeiten auf den selben Ergebnismengen. Wird einem Benutzer ein Tupelpaar aus Innerer und Äußerer Ergebnismenge präsentiert, schreitet der Index weiter fort und die nächste Anfrage richtet sich an das nachfolgende Tupelpaar. Somit wird jedes Tupelpaar zweimal bearbeitet.

Es gibt immer einen Benutzer, der als Anführer ausgezeichnet ist. Man wird Anführer, wenn man ein Tupelpaar bearbeitet, das noch niemand zuvor bearbeitet hat, d.h. die Ergebnismengen, die dieses Tupelpaar geliefert haben, sind am weitesten fortgeschritten. Nur der Anführer arbeitet auf der am weitesten fortgeschrittenen Ergebnismenge. Alle anderen arbeiten auf der Ergebnismenge, die noch nicht so weit fortgeschritten ist. Zu einem Wechsel des Anführers und der am weitesten fortgeschrittenen Ergebnismenge kommt es nur dann, wenn die Ergebnismengen gleich fortgeschritten sind und jemand der nicht Anführer war, ein Tupel bearbeitet, das niemand zuvor bearbeitet hat. Die Ergebnismenge, die dieses Tupelpaar geliefert hat, wird nun die am weitesten fortgeschrittene Ergebnismenge. Auf diese Weise wird verhindert, daß ein Benutzer ein Tupelpaar zweimal beurteilen muss.

3.2 Erkennen und Verwalten der Transitivität

3.2.1 Transitivität bei zwei Relationen

Durch die Annahme, zwei zu vergleichende Relationen sind intern duplikatfrei, kann man sich die Transitivität zu nutze machen. Findet man ein Duplikat, ist klar, dass keines der beteiligten Tupel einen weiteren Duplikatpartner finden kann. Es kommt daher zu einer Automatisierung, welche das äußere Tupel mit jedem weiteren Tupel der inneren Relation als nicht Duplikat einstuft.

3.2.2 Transitivität bei einer Relation

Jedes Tupelpaar wird zweimal bewertet. Es gibt nun zwei Möglichkeiten, daß eine Bewertung automatisiert wird:

1. Hat ein und derselbe Benutzer Duplikate so erkannt, dass aufgrund der Transitivität der Duplikatrelation ein Tupelpaar als Duplikat angenommen werden kann, fällt eine der zwei Betrachtungen dieses Tupels weg und es erfolgt eine automatische Bewertung als Duplikat. Dies schließt aber noch nicht aus, daß das Tupelpaar nicht noch von einem anderen Benutzer beurteilt wird.
2. Gibt es eine Menge von Duplikaten, die sogar zweimal als solche bewertet wurden, wird die Differenz zu ihrer transitiven Hülle für beide Betrachtungen automatisch als Duplikat bewertet, unabhängig davon, welcher Nutzer die Duplikate erkannt hatte.

Bemerkung:

Hierbei muss man sich bewusst sein, dass eine Automatisierung, die Inkonsistenzen bezüglich der Transitivität vermeiden soll, ein starkes Vertrauen in einmal durch den Benutzer getroffene Entscheidungen voraussetzt.

Algorithmus

Wird ein Element der äußeren mit allen Elementen der inneren Relation verglichen, ist das Äußere der Repräsentant einer Äquivalenzklasse, in der alle Duplikate mit diesem Tupel enthalten sind. Es gibt drei verschiedene Arten von Äquivalenzklassen:

1. Sichere Äquivalenz, mit allen Tupeln, die von zwei Benutzern als zum Repräsentant gleich beurteilt wurden.

2. Unsichere Äquivalenz, mit allen Tupeln, die bisher nur einmal Beurteilt wurden, dabei aber als Duplikat gekennzeichnet sind.
3. Konfliktäquivalenz, mit allen Tupeln, die einmal als Duplikat und einmal als etwas anderes, oder aber mindestens einmal als unbekannt gekennzeichnet wurden.

Sind alle Elemente der inneren Relation mindestens einmal mit dem Element der äußeren Relation, also ihrem potentiellen Repräsentanten, verglichen, löst der Anführer oder die Gruppe der Nachzügler das Fortschreiten in der äußeren Relation aus. Nun gibt es vier Möglichkeiten für das neue äußere Element:

1. Es ist in einer sicheren Äquivalenzklasse, d.h. es gilt als sicher, dass es sich um ein Duplikat handelt: es werden die Werte der Vergleichsergebnisse beider Benutzergruppen vom Repräsentanten übernommen (auch nicht bearbeitet).
2. Es ist in einer unsicheren Äquivalenzklasse, d.h. der Repräsentant der Äquivalenzklasse und das aktuelle äußere Element wurden bisher nur vom jetzigen Anführer bewertet. Zudem konnten alle nachfolgenden Tupelpaare ebenfalls nur vom Anführer bewertet werden. Somit gilt die Transitivität und es können die Vergleichsergebnisse des Repräsentanten mit Tupel die mit dem aktuellen äußeren Element verglichen werden sollen, übernommen werden. Die übernommenen Werte werden in der Benutzergruppe des Anführers gespeichert.
3. Es ist in einer Konfliktäquivalenzklasse: Der Konflikt muss von dem Spezialisten aufgelöst werden. Es können nur Werte übernommen werden, die auf den gleichen Bearbeiter zurückzuführen sind. Etwaige Konflikte werden wieder durch den Spezialisten behoben. Dieser nutzt ebenfalls die Transitivität, wodurch nicht zwangsläufig jeder Konflikt manuell bearbeitet werden muss.
4. Es ist in keiner Äquivalenzklasse: es kann weder bei a) noch bei b) automatisiert werden.
 - (a) Anführer: es handelt sich bisher um kein Duplikat.
 - (b) Nachzügler: es handelt sich sicher um kein Duplikat

Die Klassen werden implizit in der Extension der Arbeitsrelation gespeichert. So kann man die Zugehörigkeit eines Tupels mit dem Schlüssel id zu einer Klasse durch folgende SQL-Anfragen ermitteln:

sichere Äquivalenzklasse:

```
SELECT first_tupleID
```

```

FROM mandup.workresults
WHERE second_tupleID = id
and first_result = duplicat
and second_result = duplicat

```

unsicher Äquivalenzklasse:

```

SELECT first_tupleID
FROM mandup.workresults
WHERE second_tupleID = id
and first_result = duplicat
and second_result is null2

```

Konfliktäquivalenz:

```

SELECT first_tupleID
FROM mandup.workresults
WHERE second_tupleID = id
and (first_result = dontknow
or second_result = dontknow
or first_result = duplicat and second_result = noDuplicat
or first_result = noDuplicat and second_result = duplicat)

```

3.3 Speicherung der Daten

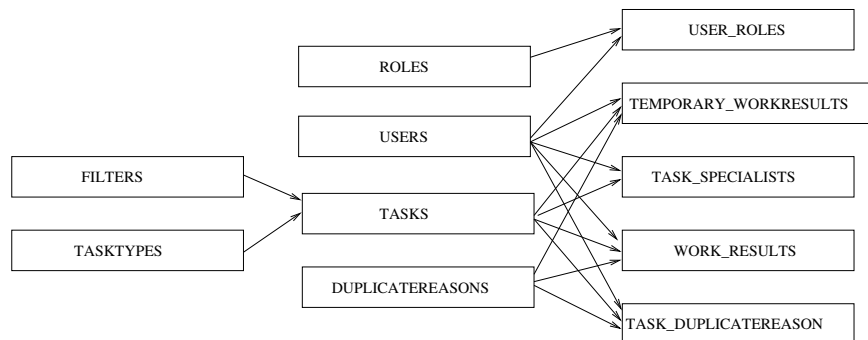


Abbildung 3.1: Relationen und Fremdschlüsselbeziehungen im Schema MAN-DUP

²Es muss in Abhängigkeit davon angefragt werden, in welcher Gruppe sich der Anführer befindet. Hier in Gruppe 1, sonst “SELECT first_tupleID FROM mandup.workresults WHERE second_tupleID = id first_result is null and second_result = duplicat”

In diesem Abschnitt werden die Relationen des verwendeten Datenbankschemas vorgestellt. Die Fremdschlüsselbeziehungen aller Relationen sind aus Abbildung 3.1 ersichtlich.

USERS Hier werden die Benutzer und ihre Passwörter gespeichert. Initial existiert ein Nutzer *admin* mit dem Passwort *adminpwd*.

ROLES Die Benutzergruppen *Administrator* und *Benutzer* werden in dieser Relation angegeben.

USER_ROLES Hier werden die Benutzer den Gruppen zugeordnet. Der Benutzer *admin* wird der Gruppe *Administrator* zugeordnet.

FILTERS In dieser Relation werden die bei der *Filterverwaltung* der Webanwendung angegebenen *Filter* gespeichert. Diese stehen dadurch beim Anlegen einer Aufgabe zur Auswahl.

TASKTYPES Die unterschiedlichen Aufgabentypen werden hier gespeichert. Initial existieren die Aufgabentypen 'Duplikatsuche ohne Transitivität in einer Relation', 'Duplikatsuche ohne Transitivität zwischen zwei sauberen Relationen' sowie 'Duplikatsuche mit Transitivität zwischen zwei sauberen Relationen'.

TASKS Hier werden die mit der *Aufgabenverwaltung* der Webanwendung angelegten *Aufgaben* gespeichert.

TASK_SPECIALIST Jeder Aufgabe wird hier ein Spezialist zugeordnet.

DUPLICATEREASONS Hier werden alle Duplikatkategorien gespeichert, die bei der Benutzung der Webanwendung angegeben werden.

TASK_DUPLICATEREASON Einer Aufgabe werden in dieser Relation unterschiedliche Duplikatkategorien zugeordnet. Beim Bearbeiten einer Aufgabe kann ein Benutzer dann zwischen Duplikatkategorien wählen, die in der Relation **TASK_DUPLICATEREASON** der Aufgabe zugeordnet worden sind.

WORK_RESULTS In dieser Relation befinden sich sämtliche zu beurteilende Tupelpaare und wenn schon vorhanden, das Ergebnis der Beurteilungen.

TEMPORARY_WORKRESULTS Diese Relation wird verwendet um abzuspeichern, welches Tupelpaar zur Beurteilung an einen Benutzer rausgereicht worden ist.

Kapitel 4

Architektur

4.1 Verwendete Technologie

Die Webanwendung wurde für auf einem Tomcat Server in der Version 5.0.28 entwickelt. Diese arbeitet mit der JSP-Spezifikation 2.0 und der Servlet-Spezifikation 2.4. Als Java Plattform wurde das JDK 1.4 benutzt. Als Datenbankmanagementsystem verwendeten wir IBM DB2 in Version 8.1.

4.2 Komponenten der Anwendung

Die Anwendung besteht aus 6 Komponenten

1. Die Taskmanagementkomponente ermöglicht es neue Aufgaben anzulegen und Bestehende zu verwalten.
2. Über die Usermanagementkomponente kann ein Administrator Nutzer anlegen und verwalten.
3. Die Loginkomponente verifiziert den Anmeldeversuch und repräsentiert bei Erfolg einen angemeldeten Nutzer.
4. Mit der Komponente Usersetting kann ein Benutzer sein Passwort ändern.
5. Mit der Filtermanagement Komponente ist es Möglich, dem Programm neue Filter bekannt zu machen.
6. Zum eigentlichen Bearbeiten einer Aufgabe dient die Work-Komponente .

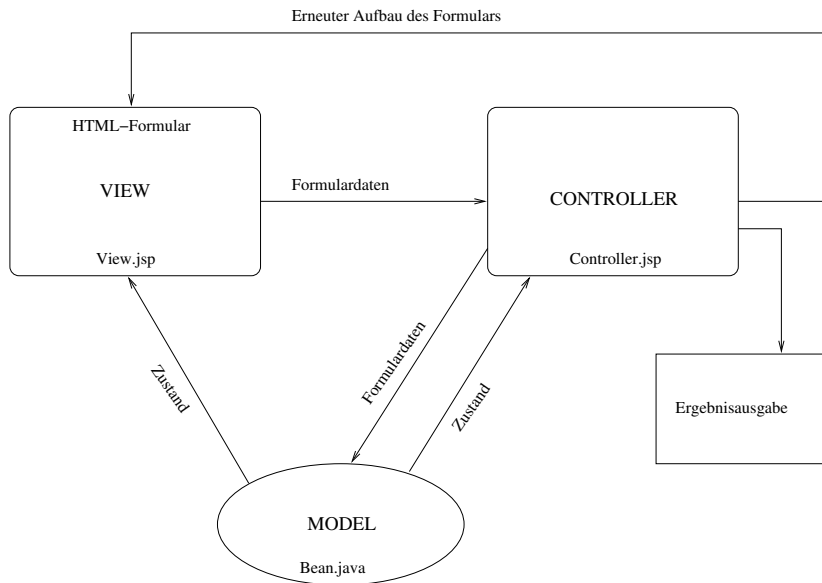


Abbildung 4.1: MVC-Model

4.2.1 Die MVC-Komponenten

Die ersten fünf der genannten Komponenten, lehnen sich an das so genannte Model-View-Controller (MVC)-Konzept an. Dieses stellt eine komfortable Möglichkeit dar, Formulareingaben zu verarbeiten. Dabei wird jede dieser Komponente aus drei Objekten aufgebaut. Das Model-Objekt, eine Java Bean, repräsentiert den Zustand des Formulars. Für jeden Parameter des Formulars hat es eine Eigenschaft mit gleichem Namen. Das View-Objekt, eine JSP Seite, erzeugt das HTML Formular und sorgt dafür, dass die Visualisierung den Zustand der Komponente widerspiegelt. Das Controller-Objekt, ebenfalls eine JSP Seite, verarbeitet die Eingaben auf dem HTML Formular und nimmt entsprechende Änderungen am Model vor. Dabei überprüft das Model-Objekt die Eingabe auf Vollständigkeit und Konsistenz, so dass das Controller-Objekt entscheiden kann, ob die Eingabe in Ordnung war, oder ob das View-Objekt neu aufgerufen werden muss, um den Client die Möglichkeit der Nachbesserung zu geben. Das Beschriebene soll noch einmal durch Abb. 4.1 verdeutlicht werden.

4.2.2 Die Work-Komponente

Diese Komponente ermöglicht das Bearbeiten von angelegten Aufgaben durch Benutzer der Webanwendung. Die beteiligten Javaklassen sowie ihr Zusammenspiel sollen im Folgenden dargestellt werden. Abbildung 4.2 gibt einen weiteren Überblick.

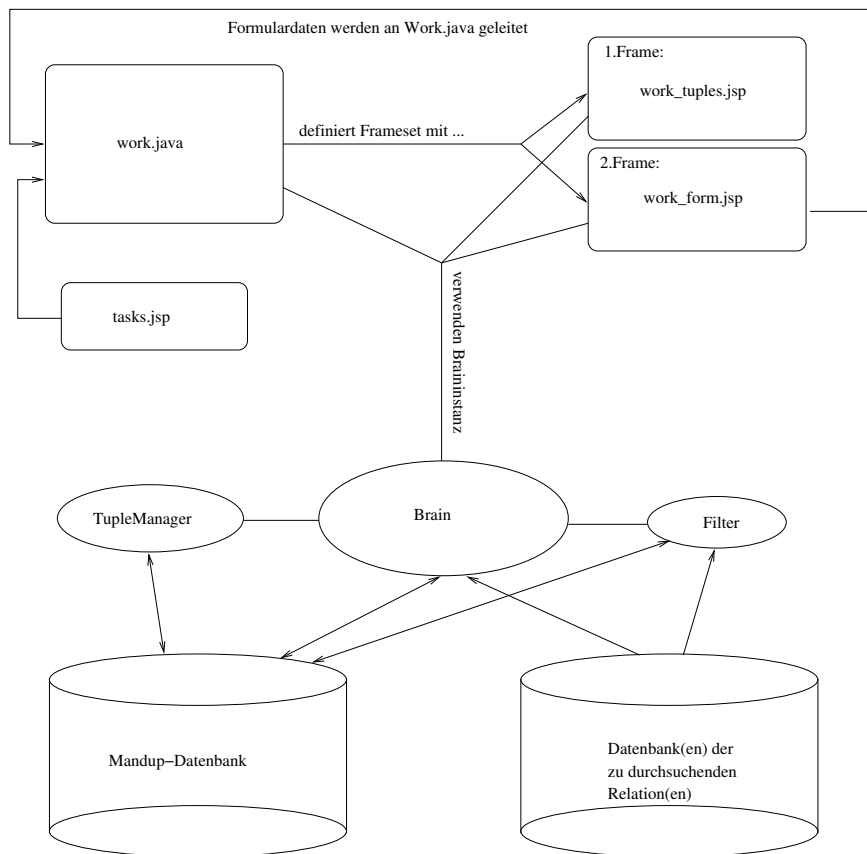


Abbildung 4.2: Das Bearbeiten von Aufgaben

4.2.2.1 Verwendete Klassen und deren Funktionalität

tasks.jsp Diese JSP-Seite ermöglicht dem Benutzer eine Aufgabe für die Bearbeitung auszuwählen. Das Ergebnis seiner Auswahl wird dann an das Servlet *work.java* gesendet.

work_tuples.jsp Diese JSP-Seite zeigt in einer Tabelle das zu beurteilende Tupelpaar an.

work_form.jsp Diese JSP-Seite erzeugt ein HTML-Formular, mit dem der Benutzer das auf *work_tuples.jsp* dargestellte Tupelpaar beurteilen kann. Bei einem *Submit* werden die Formulardaten an das Servlet *work.java* gesendet.

work.java(Servlet) Dieses Servlet definiert ein Frameset für die JSP-Seiten *work_tuples.jsp* und *work_form.jsp*. Es nimmt die Formulardaten von *work_form.jsp* entgegen und übermittelt sie an eine Instanz der Klasse *Brain*.

Ableitung von *Filter.java* In einer Ableitung der Klasse *Filter.java* wird angegeben, welche Relation(en) durchsucht werden soll(en) und welche Tupelpaare ein Duplikat darstellen können. Eine Instanz der abgeleiteten Klasse verrät die für den Zugriff auf die Tupel in den zu durchsuchenden Relationen benötigten Informationen (z.B. die VerbindungsURL(s) oder den Relationennamen).

Als geerbte Funktionalität überprüft eine Ableitung der Klasse *Filter.java* sämtliche Tupelpaare der Relation(en) und vermerkt Tupelpaare die eventuell ein Duplikat darstellen in der Relation *WORK_RESULTS*.

Ableitung von *TupleManager.java* Im *TupleManager* wird beschrieben, mit welcher Logik die in der Relation *WORK_RESULTS* vermerkten Tupelpaare zur Bearbeitung rausgereicht werden. Ein *TupleManager* stellt einen speziellen Aufgabentyp dar. Z.B. könnte eine Ableitung vom *TupleManager* die Transitivität berücksichtigen. Er arbeitet ausschließlich auf der Relation *WORK_RESULTS*.

Brain.java Diese Klasse kann man als Zentrale der Workkomponente auffassen. Ihre Funktionalität soll im folgenden Abschnitt dargestellt werden.

4.2.2.2 Zusammenspiel

Wie schon erwähnt, ist das Herzstück der Workkomponente die Klasse *Brain*. Zu jeder Aufgabe existiert genau eine Instanz dieser Klasse, welche allen Servlets und JSP-Seiten der Webanwendung zur Verfügung steht. Für die Bearbeitung einer Aufgabe *X* verwenden die Seiten *work.java*, *work_form* und *work_tuples* das Objekt *brain_X*.

Es soll nun der Ablauf des Bearbeitens einer Aufgabe skizziert werden.

1. Der Benutzer wählt als erstes aus der Seite *tasks.jsp* welche Aufgabe er bearbeiten möchte.
2. Das Ergebnis wird an *work.java* gesandt.
3. Im Servlet *work.java* wird zunächst überprüft, ob schon eine Braininstanz zur gewählten Aufgabe existiert und gegebenenfalls eine neue Instanz erzeugt.
Im Falle einer Instanzierung geschieht Folgendes im Konstruktor der Klasse *Brain*:
 - (a) Entsprechend den beim Anlegen der Aufgabe gemachten Angaben, werden Instanzen von einem *Filter* und einem *TupleManager* gebildet.
 - (b) Für den Fall, dass neue Tupel in den zu durchsuchenden Relationen sind, überprüft der Filter alle neuen Tupelpaare und schreibt Duplikatkandidaten in die Relation *WORK_RESULTS*.
 - (c) Es werden die für das Abfragen der Tupelpaare benötigten Informationen aus dem Filter ausgelesen und die Datenbankverbindung(en) hergestellt sowie SQL-Abfragen vorbereitet.
4. *work.java* generiert als HTML-Ausgabe ein Frameset für die Seiten *work_tuples.jsp* und *work_form.jsp*.
5. In *work_tuples.jsp* wird unter Angabe des Benutzers eine HTML-Tabelle, in der die zu vergleichenden Tupel dargestellt sind, bei der Braininstanz angefordert.
In der Braininstanz wird geguckt, ob für den Benutzer bereits ein zu beurteilendes Tupelpaar bekannt ist. Ist dies der Fall, so existiert ein Eintrag in der Relation *TEMPORARY_WORK_RESULTS* in dem die IDs der Tupel vermerkt sind und es muß lediglich mit HTML eine Tabelle beschrieben werden. Sollte noch kein Tupelpaar für den Benutzer existieren, so fordert die Braininstanz eins bei der TupleManagerinstanz für den Benutzer an, schreibt dieses in die Relation *TEMPORARY_WORK_RESULTS* und baut die HTML-Tabelle auf.
6. In *work_form.jsp* wird das HTML-Formular mit Unterstützung der Braininstanz aufgebaut.
7. Der Benutzer beurteilt nun das ihm präsentierte Tupelpaar mit dem Formular und das Ergebnis wird an *work.java* gesendet.
8. Das Servlet *work.java* empfängt die Formularparameter und leitet diese an die Braininstanz weiter, wo Folgendes geschieht.
 - (a) Das Ergebnis wird dem *TupleManager* bekanntgegeben. Dieser schreibt das Ergebnis in *WORK_RESULTS* und kann dann je nach Implementierung das Ergebnis für den weiteren Ablauf verwenden (Stichwort: Transitivität).

- (b) Die Braininstanz erfragt ein neues Tupelpaar für den Benutzer beim *TupleManager* und vermerkt dieses in der Relation *TEMPORARY_WORK_RESULTS*.

9. Weiter mit 4.

Dieser Fluß endet, sobald der Benutzer auf *Speichern und Beenden* klickt. Seine Beurteilung wird dann abgespeichert, aber es wird kein neues Tupelpaar angefordert.

Kapitel 5

Ausblick

In der Praxis hat sich angedeutet, dass die Transitivität für eine Relation von geringer Bedeutung ist. Da die Bearbeitung der Aufgabe den Zeitrahmen zu sprengen drohte, wurde auf die Implementierung dieser Transitivität verzichtet. Dazu müsste ein weiterer Tupelmanager geschrieben werden. Danach hätte man die Möglichkeit beim Anlegen einer neuen Aufgabe mit nur einer Relation, eine Bearbeitung mit Transitivität zu wählen.

Aus der Beschreibung ging hervor, wie ohne großen Aufwand ein neuer Filter integriert werden kann. Die Suche nach effizienterer Filterung lohnt sich, sobald große Relationen verglichen werden sollen. Ansonsten kann die Zeit, die zur Initialisierung benötigt wird explodieren. Eine weitere Möglichkeit dem langwierigen Prozess der Initialisierung zu Umgehen ist eine Partitionierung der Aufgabe. Die Möglichkeit dazu besteht derzeit lediglich auf manuellem Weg. Man kann das Problem der Initialisierung zerlegen, indem man mit kleinen Relationen anfängt, und diese schrittweise erweitert. Eine bessere Art der Partitionierung erfordert eine Veränderung bzw. Erweiterung des Programmkodes. Für beliebig große Relationen könnte man beim Anlegen der Aufgabe eine Zerlegung in der Datenbank beschreiben. Das eigentliche Filtern könnte man auf Client-Rechnern mit einem ebenfalls zu implementierenden Java-Programm ausführen. Vorteile dieser Variante sind, dass die Initialisierung keine Last auf dem Webserver erzeugt, dass man mehr Rechenleistung zu Verfügung stellen kann, als auf dem Web-Server vorhanden ist, und dass man durch eine ausreichend feine Zerlegung nicht vor dem Problem steht, einen Rechner mehrere Tage zu blockieren.

Als eine zusätzliche Erweiterung könnte man eine Komponente implementieren, die das gewonnene Ergebnis der manuelle Duplikatsuche mit dem eines Algorithmus abgleicht. Bibliotheken für die grafische Darstellung des Vergleiches wären einfach einzubinden und eine Visualisierung sicher sehr komfortabel.