Konzeption einer datenbankbasierten Plattform für die Informationsfusion

Roland Jesse Ingolf Geist
Oliver Dunemann

Fakultät für Informatik, Universität Magdeburg
Postfach 4120, D-39016 Magdeburg
{jesse@isg,geist@iti,dunemann@iti}.cs.uni-magdeburg.de

Zusammenfassung

Informationsfusion als Prozess der Integration und Interpretation heterogener Daten mit dem Ziel der Gewinnung neuer Informationen einer höheren Qualität eröffnet eine Vielzahl von Anwendungsgebieten. Gleichzeitig erfordert dieser Prozess aber auch eine enge Verzahnung der bislang häufig noch isoliert vorliegenden Werkzeuge und Techniken zum Zugriff auf heterogene Datenquellen, deren Integration, Aufbereitung, Analyse und Visualisierung. In diesem Konzept werden erste Ergebnisse der Entwicklung einer Workbench vorgestellt, die durch konsequente Nutzung von Datenbanktechniken eine durchgängige Unterstützung dieser Schritte ermöglicht.

Dabei wird durch einen modularen Aufbau insbesondere darauf Wert gelegt, dass das System an verschiedene Anforderungen angepasst werden kann. So lassen sich jederzeit neuartige Datenquellen erschließen, indem entsprechende Adapter eingebunden werden. Die auf den Informationen ausführbare Funktionalität lässt sich durch anwenderdefinierte Operatoren und Prozeduren erweitern, während Bearbeitungs- und Auswertungskomponenten zur Laufzeit dynamisch nachgeladen sowie aus dem System entfernt werden können. Das Ziel ist ein schlanker Kern, der anwendungsspezifisch mit der benötigten Funktionalität ergänzt werden kann.

Diese Arbeit wird gefördert von der DFG (FOR 345/1).

Inhaltsverzeichnis

1 Einleitung

1.1 Informationsfusion – Begriff und Anwendungen

Der Begriff der *Informationsfusion* beschreibt den Prozess der Integration und Interpretation von Daten aus heterogenen Quellen sowie die darauf aufbauende Konstruktion von Modellen für einen bestimmten Problembereich mit dem Ziel der Gewinnung von Informationen einer neuen, höheren Qualität. Diese Definition erklärt die Informationsfusion als ein interdisziplinäres Gebiet, das auf Methoden und Techniken verschiedener Bereiche, wie z.B. Datenbanken, Statistik, Maschinellem Lernen und Visualisierung zurückgreift [DGJ+01].

Der Fusionsprozess beinhaltet dabei die verschiedenen Aspekte Datenzugriff, Datenintegration, Analyse und Verdichtung, Präsentation und Weiterverarbeitung sowie der Repräsentation der Metadateninformationen (vgl. Abbildung 1.1). Eine genaue Anforderungsanalyse und Beschreibung der einzelnen Bereiche sind in [CSS99] dargestellt.

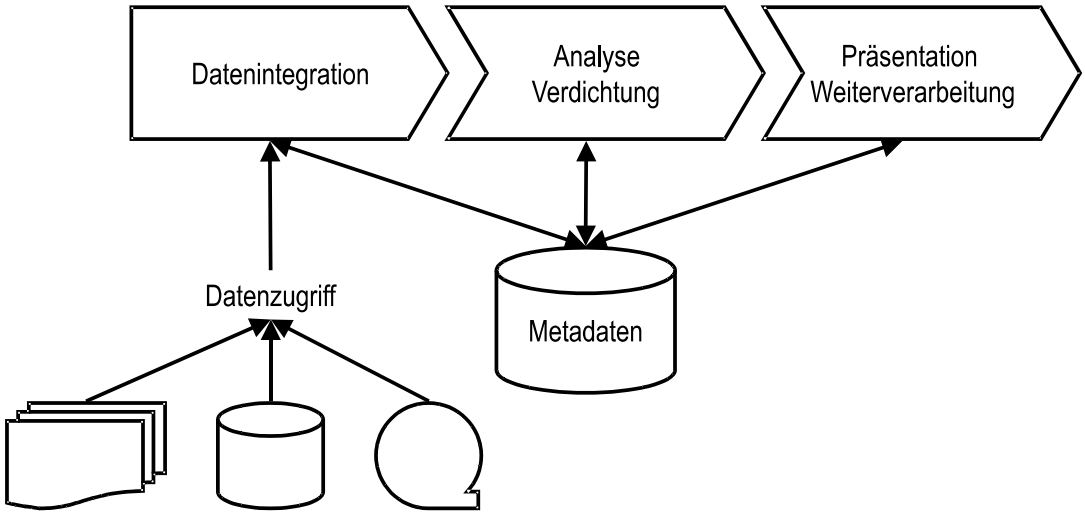


Abbildung 1.1: Aspekte eines Prozesses zur Informationsfusion

Die verschiedenen Schritte der Integration und Analyse der Daten können durch einen Graphen modelliert werden. Hierbei beschreiben die Knoten des Graphen die Datenquellen und die Operationen auf diesen Quellen. Die Kanten beschreiben die Aufeinanderfolge der Operationen. Dieser Graph dient im Weiteren als Modell für ein *Worksheet*, welches die verschiedenen Sichten auf den Prozess beschreibt. Ein solcher Graph kann

entweder direkt grafisch erstellt oder implizit durch die Anwendung der verschiedenen Operationen auf die Daten mit einem Wizard-Dialog erzeugt werden.

Es existieren vielfältige Anwendungsgebiete für die Informationsfusion. Die Untersuchung von Gensequenzen aus verschiedenen Gen- und Stoffwechseldatenbanken in der Bioinformatik ist ein Beispiel hierfür. Weiterhin ist der Produktionsvorbereitungsprozess in der Giessereiindustrie ein Anwendungsgebiet. Ein weiteres Beispiel stellt die Analyse von Konto- und Kundendaten zur Bonitätsprüfung dar. Aus diesem Bereich wurde auch das Beispielszenario gewählt, welches in diesem Papier zur Verdeutlichung herangezogen wird. Abbildung 1.2 zeigt einen Graphen, der einen Prozess zur Berechnung von Kreditrisiken in einer Bank darstellt.

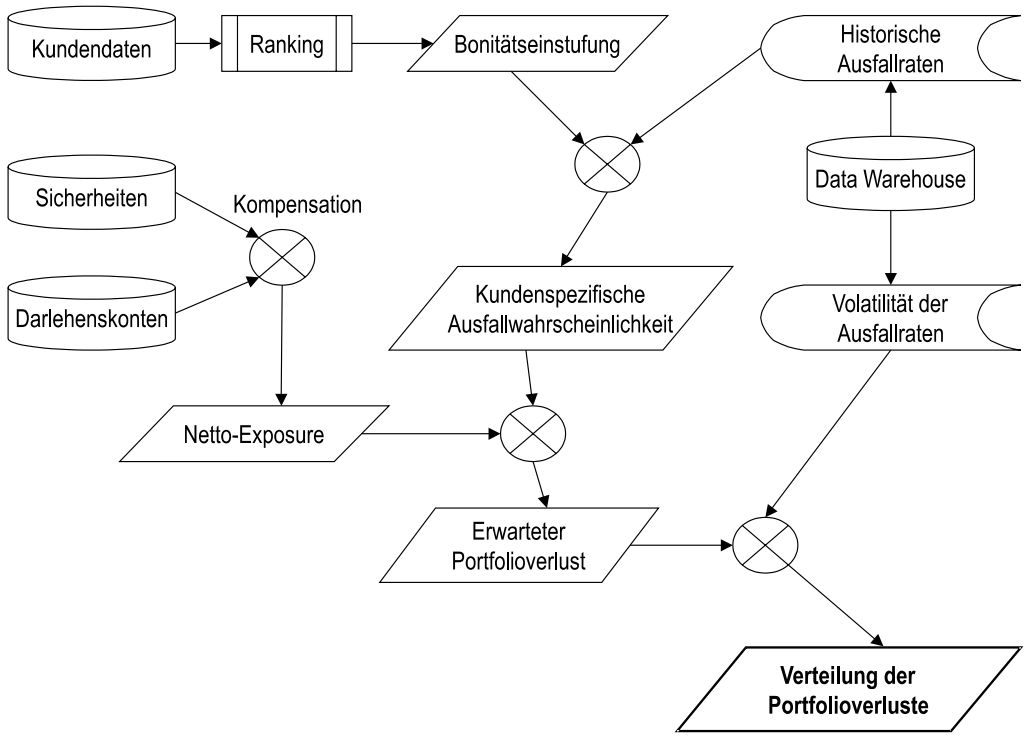


Abbildung 1.2: Beispiel eines Graphen eines Fusionsprozesses

Die Bedeutung der Aufgabe, verteilt und heterogen vorliegende Datenbestände in einer integrierten Form darzustellen und zu analysieren, wurde in letzter Zeit zunehmend erkannt. Die wissenschaftliche Arbeit konzentrierte sich dabei zunächst auf das Erarbeiten von Lösungen für Teilschritte. So wurden die technischen Voraussetzungen zum Zugriff auf verteilte, heterogene Datenbestände und Methoden für die Integration auch über Paradigmengrenzen hinaus geschaffen. Werkzeuge für spezielle Nachbearbeitungs- und Analyseschritte wie beispielsweise Data-Mining oder Visualisierung wurden entwickelt. Da jedoch die Integration dieser Komponenten bisher nicht oder nur teilweise durchgeführt wurde, konnte erhebliches Optimierungspotential nicht ausgeschöpft werden. Hier ist u.a. eine dynamische und anpassungsfähige Entscheidungsfindung bezüglich der Materialisierung von Zwischenergebnissen zu berücksichtigen.

Die Basis der Informationsfusion bildet eine enge Verzahnung der Schritte Integration heterogener Daten, deren Aufbereitung und Analyse. Nur so kann eine interaktive Arbeitsweise unterstützt werden, die dem iterativen Charakter des Fusionsprozesses gerecht

wird. Hierfür wird ein Vorrat an integrierten Werkzeugen für die einzelnen Schritte des Prozesses benötigt. Im Folgenden werden erste Ergebnisse der Entwicklung einer solchen *Workbench* vorgestellt.

Durch ein offenes und modulares Konzept wird ein Rahmen aus Basisdiensten geschaffen, auf dem aufbauend weitere Komponenten entwickelt werden können. Dabei werden bereits in der Analysephase Aspekte der Teilaufgaben der Informationsfusion berücksichtigt, indem von Praxisbeispielen ausgehend beispielhaft Fusionsprozesse modelliert werden. Solche Basisdienste stellen neben den Integrations- und Darstellungskomponenten einen zentralen Authentifizierungsmechanismus und eine einheitliche Fehlerbehandlung zur Verfügung.

1.2 Aufbereitung und Integration der Daten

Nach der Einleitung und der Darstellung des Problemfeldes soll nachfolgend auf die Grundlagen der Datenanalyse sowie die Aufbereitung und Integration der heterogenen Daten eingegangen werden. Hierdurch soll die Qualität der Daten gesichert werden, da sich nur aus qualitativ hochwertigen Daten relevante Analyseergebnisse ableiten lassen.

Im Data Warehouse Bereich werden Data Cleaning bzw. ETL-Werkzeuge zur Aufbereitung und Integration der Daten benutzt. Um diese Aktivitäten zu ermöglichen, muss ein Werkzeug folgende Eigenschaften aufweisen:

- **Schnelle Reaktionszeiten**: Die Erkennung und Lösung von Konflikten erfordert eine Interaktion mit dem Anwender. Um diese Arbeitsweise zu ermöglichen, müssen die Werkzeuge Ergebnisse der Aufbereitungsschritte schnell an den Anwender ausgeben, so dass dieser sie in einem frühen Stadium beurteilen kann. Somit ist die Anwendung von langlaufenden Batch-Prozessen, die auf dem gesamten Datenbestand arbeiten, nicht möglich. Vielmehr sollten zunächst Stichproben untersucht werden, um anschließend die Ergebnisse auf den gesamten Datenbestand anzuwenden.

- **Integration der Werkzeuge**: Für die Aufbereitung der Daten müssen unterschiedliche Aktionen und Algorithmen zur Konfliktentdeckung und -lösung angewendet werden. Da sich bei diesem Prozess Konflikte gegenseitig bedingen können, und somit nicht sofort zu erkennen sind, ist eine Integration der Werkzeuge in einem System notwendig, wodurch die Erkennung und Lösung dieser verschachtelten Konflikte erst ermöglicht wird.

- **Durchgehende grafische Benutzerführung**: Die Interaktion mit den Werkzeugen soll nach Möglichkeit durch eine grafischen Benutzerführung erleichtert werden. Hierdurch können einerseits Einarbeitungszeiten in komplexe Programmierumgebungen verringert werden, und andererseits wird auch die Entdeckung und Lösung von Konflikten während der Aufarbeitung und Integration der Daten erleichtert. Zur Unterstützung der Iteration im Integrationsprozess ist eine Undo-Funktion von zentraler Bedeutung. Somit sollten alle Aktionen zunächst virtuell ablaufen und nicht sofort materialisiert werden.

Aus diesen Überlegungen ergibt sich ein interaktiver Ansatz der Datenaufbereitung und -integration. Dabei wird davon ausgegangen, dass das globale Schema bereits vorliegt und die lokalen Daten auf dieses abgebildet werden müssen. Für diese Abbildung werden die Anfrage- und Integrationfähigkeiten der Multidatenbanksprache FRAQL eingesetzt und erweitert. Diese ist mit ihren Eigenschaften in [SCS00] ausführlich beschrieben.

Zunächst erfolgt die Anwendung der Integrationsschritte auf einer Stichprobe des gesamten Datenbestandes. Diese kann durch bekannte Sampling-Verfahren wie z.B. [Vit87] erzeugt werden, die wie unter anderem in [OR86] und [CMN99] beschrieben, in das Multidatenbanksystem integriert werden. Die nachfolgende Beschreibung zeigt, wie die einzelnen Konfliktklassen behandelt werden. Dazu ist zu sagen, dass die Sicht auf die Daten relation bzw. objektrelational ist. Dieses wird durch die Multidatenbanksprache im Kern der Fusion Engine mit Hilfe von Adaptern zu den einzelnen Datenquellen gewährleistet. Dadurch können Operatoren generisch modelliert werden, da sie stets auf dem selben Datenparadigma arbeiten, selbst, wenn es sich bei den Daten tatsächlich beispielsweise um eine Textdatei handelt.

Zunächst müssen die lokalen Schemata auf das globale Schema abgebildet werden. Dieses erfolgt mit Umbenennungen, Hinzufügen und Löschen von Attributen. Hierbei werden die typischen Möglichkeiten der Multidatenbanksprache FRAQL benutzt. Weiterhin können Metadatenkonflikte auftreten. Diese zeichnen sich dadurch aus, dass ein Teil der Daten in den Metadaten, wie z.B. den Attributnamen, und ein Teil in den Datenwerten modelliert ist. Eine Lösung dieser Konflikte ist durch die Benutzung von Metadaten in den Anfragen gegeben. Weitere Informationen zu diesem Bereich finden sich u.a. in [Sch98].

Nach der oben beschriebenen Anpassung der Schemata werden Konflikte in den Datenwerten betrachtet. Diese sind allerdings nicht unabhängig von der Schemaanpassung zu sehen, da eine Abhängigkeit in beide Richtungen besteht. Zur Aufbereitung der Daten können arithmetische Ausdrücke oder auch Zeichenkettenfunktionen angewendet werden. Hiermit können Beschreibungskonflikte gelöst werden. Sind diese Mittel nicht ausreichend, kann der Anwender eigene benutzerdefinierte Funktion auf die Daten anwenden. Hier wird der Vorteil der einfachen Anwendbarkeit der Funktionen zugunsten einer erhöhten Flexibilität aufgegeben. Alle diese Operationen können zunächst auf einer Stichprobe ausgeführt werden, da hier bereits bestimmte Zusammenhänge schnell erkannt werden können. Weiterhin fließen alle Operationen zunächst in eine Sichtdefinition ein, sind also effizient zu modifizieren. Da die Fusion Engine die Entscheidung über die Materialisierung von Zwischenergebnissen trifft, ist eine transparente Implementierung der Undo-Funktion möglich.

Als grafische Unterstützung stehen neben der Tabellenansicht mit der Möglichkeit der Anwendung der Operationen weitere Ansichten zur Verfügung, die im Abschnitt 1.4 beschrieben werden.

Dieser Ansatz der datenorientierten Aufbereitung und Integration der heterogenen Quellen unterstützt die iterative und interaktive Lösung der auftretenden Konflikte. Durch die anfängliche Benutzung von Stichproben kann eine schnelle Reaktionszeit des Systems erreicht werden. Allerdings ist es so unmöglich, alle Ausreisser bzw. Fehler in den Daten zu finden, hierzu muss weiterhin die gesamte Datenmenge betrachtet werden.

Hat der Anwender unter Verwendung der vorgestellten Integrationsmechanismen die Daten seinen Anforderungen entsprechend aufbereitet, existieren zwei Möglichkeiten der

weiteren Verwendung des Ergebnisses. Zum Einem kann die erzeugte globale Sicht auf die Daten direkt weiterverwendet werden, zum Anderen kann eine Materialisierung in eine lokale Datenbank vorgenommen werden, um weitere Bearbeitungs- und Analyseschritte der Daten zu beschleunigen.

1.3 Datenanalyse

In vielen Anwendungsfällen liefert die Integration verschiedener Datenquellen allein noch keinen Gewinn. Gerade bei einer größeren Anzahl von Quellen bleiben auf Grund des resultierenden Datenvolumens interessante Aspekte oft verborgen. Daher sind die integrierten Daten weiter zu analysieren, um etwa Muster, Tendenzen oder Regelmäßigkeiten aufzudecken. Zu diesem Zweck wurden in der Vergangenheit insbesondere auf dem Gebiet des Data Mining eine Vielzahl von Verfahren entwickelt [FPSSU96]. In Verbindung mit Zugriffs- und Integrationsmechanismen für heterogene Datenquellen versprechen diese Techniken neue, vielfältige Einsatzmöglichkeiten.

Ein Defizit aktueller Ansätze zur automatischen Datenanalyse in großen Datenbeständen – im Vergleich zu OLAP-Anwendungen, die eher eine benutzergesteuerte, navigierende Form darstellen – ist die unzureichende Kopplung zum Datenbanksystem. So arbeiten viele Data-Mining-Tools hauptspeicherbasiert und sind damit hinsichtlich der zu untersuchenden Datenmenge beschränkt. Andererseits bieten auch moderne Datenbankmanagementsysteme(DBMS) kaum Unterstützung in Form spezieller Operatoren oder Optimierungsstrategien. Ausnahmen sind hier u.a. [CDH+99]. Trotzdem bietet eine enge Kopplung eine Reihe von Vorteilen, wie die Nutzung der durch das DBMS bereitgestellten Zugriffsstrukturen und Optimierungsstrategien, der Speicherverwaltung für die Bearbeitung großer Datenmengen sowie der ausgereiften Parallelisierungsmechanismen moderner Systeme [Cha98].

Vor diesem Hintergrund wird im Rahmen der hier vorgestellten Workbench eine enge Verbindung zwischen Analysetechniken und Datenbankfunktionalität angestrebt. So werden die einzelnen Analyseoperationen als SQL-Programme ähnlich gespeicherten Prozeduren implementiert und in der Fusion Engine ausgeführt. Auf diese Weise lassen sich einzelne SQL-Anweisungen als Teil einer Analyseoperation direkt auf den Quelldaten bzw. auf den (gegebenenfalls materialisierten) Ergebnisrelationen anwenden.

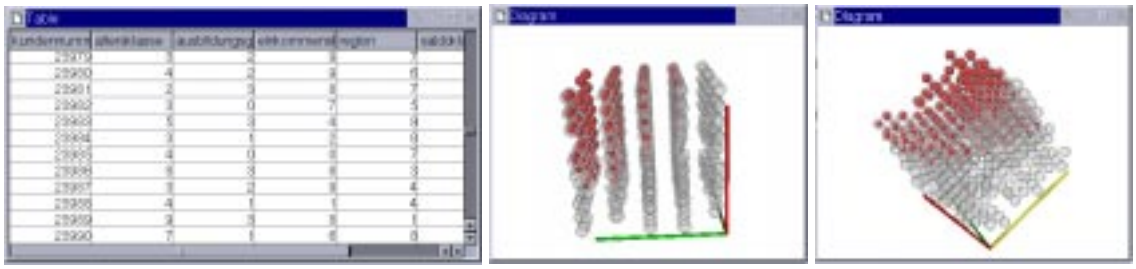
Die Umsetzung von Data-Mining-Verfahren auf der Basis von SQL ist eine aktuelle Herausforderung. Erste Arbeiten beschäftigen sich im Wesentlichen mit Klassifikationsverfahren [CFB99] und der Ableitung von Assoziationsregeln [STA98]. Dabei wurde deutlich, dass noch Performance-Probleme bestehen, die durch neue Datenbankprimitive und Optimierungstechniken zu lösen sind [Cha98].

Neben diesem einfachen Klassifikationsverfahren ist die Umsetzung weiterer Data-Mining-Operationen auf Basis von SQL geplant, genannt seien an dieser Stelle der Apriori-Algorithmus zur Aufdeckung von Assoziationsregeln und Clustering-Verfahren.

1.4 Visualisierung zur Benutzerunterstützung

Das Verständnis großer, meist multidimensionaler Datenbestände und die Extraktion von Informationen daraus setzt ihre umfassende Exploration voraus. Das Erkennen von Mus-

tern und Trends, die Navigation im Datenbestand sowie das Auffinden von Beziehungen zwischen einzelnen Datensätzen erweist sich hierbei schnell als schwierig, oftmals als unmöglich [Eic00]. Es ist somit Aufgabe einer die Exploration unterstützenden Visualisierung, große Datenmengen handhabbar zu gestalten, die Betrachtung sowie Manipulation von Objekten in der Datenbasis zu ermöglichen und die Darstellung so aufzubereiten, dass Information durch den Benutzer leichter auffindbar wird. Diese Notwendigkeiten werden dadurch unterstrichen, dass die Interaktion gemäß den Abschnitten 2 und 1.2 einen wichtigen Aspekt im Prozess der Informationsfusion darstellt. Integrations- und Fusionsoperationen sind benutzergesteuert auszuführen beziehungsweise zu spezifizieren, Anfragen zu formulieren und die Ergebnisse auszuwerten, um gegebenenfalls den Fusionsprozess wieder zu revidieren.



(a) Tabellendarstellung

(b) Separierung von Bonitätsklassen

(c) Einfluss selektiver Attribute auf die Bonität

Abbildung 1.3: Verschiedene Sichten eines exemplarischen Datenbestandes

In Abbildung 1.3 sind drei mögliche Sichten auf den selben Datenbestand dargestellt. In Bild 1.3(a) erfolgt die Darstellung in Tabellenform während die Bilder 1.3(b) und 1.3(c) eine Filterung mittels Gaussian Splatting widerspiegeln. Beide Darstellungsformen ergänzen sich dabei. Eine Tabellenansicht für eine gegebene, beliebige Relation ist verhältnismäßig einfach zu erstellen. Ihre Spalten werden mit den Relationsattributen populiert und in den Zeilen respektive die zugehörigen Werte eingetragen. Ergebnisse sind auf diese Weise schnell dem Anwender anzuzeigen. Etwas aufwendiger gestaltet sich prinzipiell die Erzeugung der visuell reicheren Diagramme. Sie sind nicht immer vollständig automatisch aus einer gegebenen Relation abzuleiten. Einzelne Dimensionen der Eingabedaten sind untereinander variabel ins Verhältnis zu setzen. Sinnvolle Kombinationen in Hinsicht auf die Exploration und Interpretation durch den Anwender ergeben sich dabei oftmals erst aus dem Kontext und sind somit nur schwerlich zu generalisieren.

Die Motivation der hier dargestellten Diagramme ist ein besseres Verständnis der Beziehungen einiger Attribute von Kunden zu ihrer Bonität. Dargestellt wird der Einfluss auf selbige von Seiten der Region (entlang der grün eingefärbten x-Achse), des Einkommens (entlang der rot gefärbten y-Achse) und des Ausbildungsgrades (entlang der gelb gefärbten z-Achse). Weitere Attribute fließen in die Betrachtung an dieser Stelle nicht mit ein, können und sollten allerdings Gegenstand weiterer Untersuchungen sein. Grau dargestellt ist der gesamte Bereich der Eingaberelation, welche die Trainingsdaten enthält. Rot und somit kontrastiv dunkler hervorgehoben ist eine Sicht auf den gleichen Eingabebereich skaliert entsprechend der Bonitätsklasse. Die Abbildung 1.3(c) offenbart den verhältnismäßig starken Einfluss der Region auf die Bonität, welcher sich darin

ausdrückt, dass die rot (dunkler) dargestellten Bereiche sich mit größerem Abstand vom Koordinatenursprung maximieren. Dieser Effekt ist in abgeschwächter Form auch entlang der roten Achse zu beobachten, womit der stark gewichtete, wenn auch lineare Einfluss der Einkommensklasse auf die Bonität deutlich wird. Eine nahezu ausgeglichene Gleichverteilung ist für den Einfluss des Ausbildungsgrades abzulesen. Im Vergleich zu den anderen beiden Variablen wirkt er sich am schwächsten auf die Bonität eines Kunden aus. Abbildung 1.3(b) ist die Aufteilung der einzelnen Kundendatensätze auf die unterschiedlichen Bonitätsklassen zu entnehmen. Deutlich wird die starke Belegung der fünf am stärksten populierten Klassen. Die wenigen Ausreisser, die sich in den anderen Klassen wiederfinden, gehen in der Darstellung etwas unter.

Die Diagramme aus Abbildung 1.3(b) und 1.3(c) sind mit Methoden des Volumenrenderings erzeugt. Die Transformation von multidimensionalen Datensätzen mit verschiedenen Attributen für jeden einzelnen Datenwert in eine Volumenstruktur und deren anschließende Visualisierung ist eine verhältnismäßig einfache Möglichkeit, derart strukturierte Daten zur visuellen Exploration aufzubereiten. Abbildung 1.4 spiegelt das der

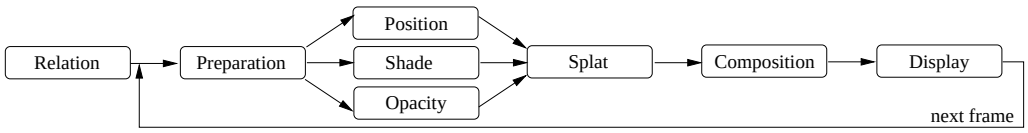


Abbildung 1.4: Visualisierungspipeline

Volumendarstellung zugrunde liegende Modell der Visualisierungspipeline wider. Die Relation als Ausgangspunkt repräsentiert hierbei generell in einer Datenbank gespeicherte Daten als Quelle für die weitere Verarbeitung. Diese werden für die Darstellung aufbereitet. In diesem Vorbereitungsschritt erfolgt die Zuordnung einzelner Attribute zu den Raumdimensionen, passende Farbgebungen sowie die Bestimmung der Durchlässigkeit darzustellender Voxel. Weitere Präsentationsvariablen können während dieser Phase einfach in die Darstellung eingebracht werden. In der mit Splat bezeichneten Phase wird den einzelnen Voxeln jeweils ein 3D-Gauss-Filter zugeordnet, der somit für die Abbildung der erzeugten Rauminformationen auf die Darstellungsfläche sorgt. Während der Komposition werden die unterschiedlichen Präsentationsinformationen zusammengefasst und respektive Grafikobjekte erzeugt, woraufhin das eigentliche Rendering und somit die Ausgabe auf dem Bildschirm angestoßen wird. Der Prozess wiederholt sich für den gesamten Eingabedatensatz.

Dieses Vorgehen basiert im wesentlichen auf dem Pipeline-Konzept der VTK-Bibliothek [SML98]. Dieses baut auf dem Prinzip der „lazy evaluation“ auf. Dabei wird ein Schritt der Pipeline nur dann ausgeführt, wenn von ihm generierte Daten zur weiteren Berechnung benötigt werden. Eine Modifikation der Eingaberelation bewirkt somit nur dann eine Neuberechnung der Szene, wenn eine Aktualisierung der Darstellung oder die Abfrage von Zwischenergebnissen der Pipeline erforderlich wird. Unnötige Berechnungen werden hierdurch effektiv vermieden.

1.5 Ziele des Projektes

Neben der Erstellung und Erweiterung der Basisdienste und der Entwicklung einer Bibliothek von Operatoren ist geplant, Erkenntnisse verwandter Forschungsgebiete in die Workbench einfließen zu lassen. Denkbar sind hier Lernverfahren zur Optimierung einzelner Prozessschritte oder sogar des Gesamtprozesses, sowie Methoden der Wissensakquisition zur Einbindung natürlichsprachlicher Texte in den Fusionsprozess.

Zur Vervollständigung wird in erster Linie die Integration weiterer Fusionsoperatoren angestrebt. Neben datenbankorientierten Optimierungs- und Analysefunktionen sind hier insbesondere interaktive Data Mining-Verfahren sowie Methoden zum automatischen Lernen interessant. Erstere ermöglichen beispielsweise mittels Methoden des interaktiven Clusterings eine weiterführende Benutzerunterstützung, während letztere die Analyse dahingehend unterstützen, dass sie wiederkehrende Verhaltens- und Abhängigkeitsmuster in größeren Datensätzen zu entdecken helfen. Im Zuge der Implementierung von Operatoren findet zum Einen ein Vergleich verschiedener Implementierungen des selben Verfahrens statt, um für ein gegebenes Szenario den nach Möglichkeit laufzeitoptimalen Algorithmus bestimmen zu können. Zum Anderen werden Datenbankprimitive identifiziert, die bestimmte häufiger benötigte Formen von Anfragen unterstützen. Diese werden in das FRAQL-System intregriert. Beispielsweise können Klassifizierungsalgorithmen oft auf statistischen Daten (Histogrammen) über die Basisdaten ausgeführt werden. Dadurch, dass somit auf vorverdichteten Informationen gearbeitet wird, ist die zu betrachtende Datenmenge kleiner. Hierdurch können Performancegewinne erzeugt werden. Diese müssen wiederum in Relation zur Erstellungszeit und des Speicherbedarfs der Statistiken betrachtet werden. Da die Statistiken aber für mehrere Auswertungen Anwendung finden können, ist hier eine Abwägung zu treffen.

Zusätzlich zur Einbettung weiterer Fusionsoperatoren in die Workbench ist eine Erweiterung der möglichen grafischen Sichten auf bestehende, generierte sowie abzuleitende Informationsbestände derzeit in der Entwicklung. Bezüglich der Diagrammdarstellungen, die der Unterstützung von Beziehungsdarstellungen einzelner Dimensionen der zugrunde liegenden Daten dienen, um Kausalitäten zwischen Datensätzen erkennbar zu gestalten ist die Erstellung automatisierter Filter von besonderem Interesse. Diese dienen der Aufbereitung des Eingabestroms innerhalb der Visualisierungspipeline, so dass die manuelle Darstellungsbearbeitung auf ein geringes Maß reduziert wird.

2 Architektur

Eine Workbench zur Unterstützung der Informationsfusion muß eine effiziente und interaktive Analyse grosser, zum Teil heterogener Datenbestände ermöglichen. Dieses umfasst die Definition und Ausführung von Anfragen, die Transformation von Daten sowie die Anwendung von Analyseoperationen und die Visualisierung der Ergebnisse. Vergleichbare Anforderungen sind auch in OLAP-Umgebungen zu finden, so dass für die Fusionsworkbench ein Architekturansatz gewählt wurde, der den Architekturen von OLAP-Anwendungen ähnlich ist. Abbildung 2.1 zeigt eine Sitzung, bei der zwei grafische Auswertungstools (z.B. ANGIE) sowie ein Batch-Prozess mit der Engine verbunden sind.

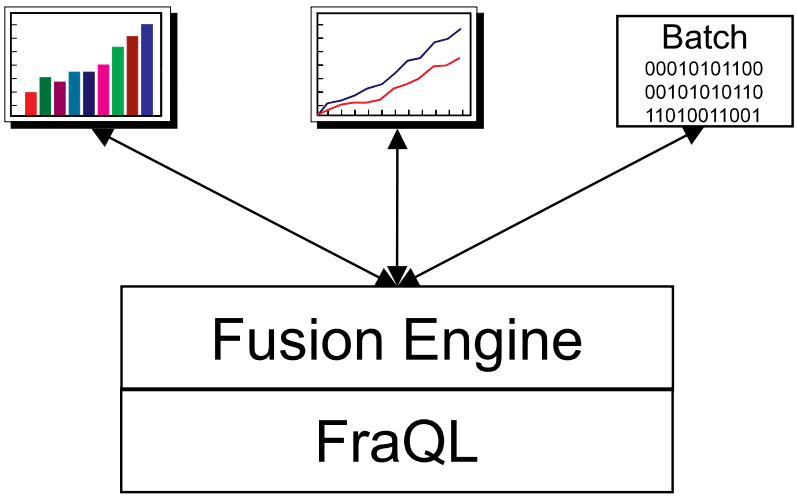


Abbildung 2.1: Verbindung von drei Clients mit der Fusion Engine

Die Basis des Systems bildet die *Fusion Engine*, die auf dem Anfragesystem FRAQL für Multidatenbanken aufbaut. Dieses Anfragesystem ermöglicht einen transparenten Zugriff auf verschiedene Datenquellen und stellt Mechanismen zu deren Integration bereit [SCS00]. Die Engine umfasst weiterhin eine lokale Datenbank für temporäre Daten (z.B. Materialisierungen und Ergebnisse) sowie Metadaten. Die eigentlichen Fusionsoperatoren, die ähnlich gespeicherten Prozeduren direkt auf den integrierten Datenbeständen ausgeführt werden können, werden über einen Operator-Manager in die Fusion Engine eingebunden. Der Worksheet-Manager verwaltet komplette Fusionsprozesse jeweils in einem Worksheet. Dieses enthält die Abhängigkeiten und Zustände einzelner Aufbereitungs- und Analyseschritte. So müssen bei Daten- oder Parameteränderungen nur die betroffenen Schritte neu ausgeführt werden. Zusätzlich können nicht voneinander abhängige ausführbare Operationen (es existieren keine Vorgänger oder alle Vorgängeroperationen sind bereits erfolgreich durchgeführt worden) parallel durchgeführt werden. Tritt in ei-

nem Zweig des Graphen ein Fehler auf, der einen Abbruch an der entsprechenden Stelle zur Folge hat, können dennoch davon nicht abhängige Zweige weiter ausgeführt werden. Eine genauere Beschreibung der Zustände von Knoten wird in Abschnitt 3.4.7 geliefert.

Die Benutzungsschnittstelle wird durch das Workbench-Frontend bereitgestellt. Zunächst wird hierzu ein grafisches Analyse- und Definitionswerkzeug implementiert, mit dem der Anwender über die Fusion Engine Zugriff auf die Daten der einzelnen Quellen hat. So können interaktiv Integrations- und Fusionsoperationen ausgeführt, Anfragen formuliert und die Ergebnisse visualisiert werden. Auch das Erstellen und Verwalten von Worksheets kann von einem Anwender mit Administratorrechten mit Hilfe dieses Tools durchgeführt werden.

Die Architektur der Workbench ist mit der Trennung in Fusion Engine und Frontend mit Ansätzen aus dem OLAP-Bereich vergleichbar. Ein wesentlicher Unterschied besteht jedoch darin, dass die zu analysierenden Daten nicht vorab extrahiert, transformiert, bereinigt und redundant in einem Data Warehouse abgelegt werden müssen. Statt dessen ermöglicht die Verwendung eines Multidatenbank-Anfragesystems innerhalb der Fusion Engine den transparenten Zugriff auf die Quellen und die Anwendung von Transformations- und Integrationsoperationen. Auf diese Weise können erste Analysen durchgeführt werden, ohne dass zuvor Daten aufwendig migriert und transformiert werden müssen. So lassen sich relevante Datenausschnitte selektieren und Operationen parametrisieren. Für die tiefergehende Analyse können die Ergebnisse einzelner Schritte anschließend materialisiert werden, um so eine effiziente Ausführung zu erreichen.

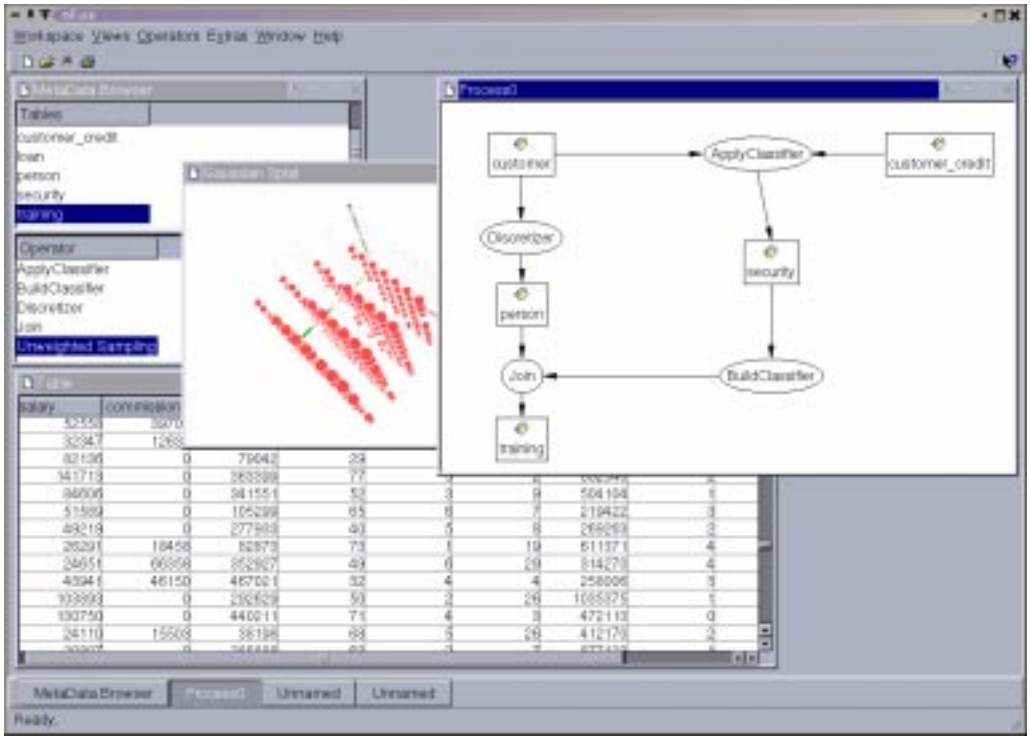


Abbildung 2.2: Prototyp mit Darstellung eines einfachen Prozessgraphen sowie verschiedenen Sichten auf einen exemplarischen Datensatz

Abbildung 2.2 zeigt eine Beispielsitzung mit der Workbench. Der dargestellte Prozessgraph zur Bestimmung der Abfolge einzelner Fusionsschritte wird interaktiv aufge-

baut. Eine Übersicht über die verfügbaren Datenquellen (Relationen, Sichten, etc.) sowie Operatoren liefert der Metadaten-Browser, im Bild links oben dargestellt. Exemplarisch sind als zwei Sichten auf die Relation mit den Trainingsdaten eine Tabelle und ein einfaches Diagramm abgebildet (siehe dazu auch den vorherigen Abschnitt 1.4).

Das System wird auf verschiedenen Unix-Plattformen implementiert. Die Verwendung einer standardisierten Datenbank-API (ODBC), des Visualization Toolkits [SML98] als Basis der Visualisierungskomponente sowie von Qt als Grundlage für die Benutzungsschnittstelle gewährleistet eine potenziell weiterführende Plattformunabhängigkeit.

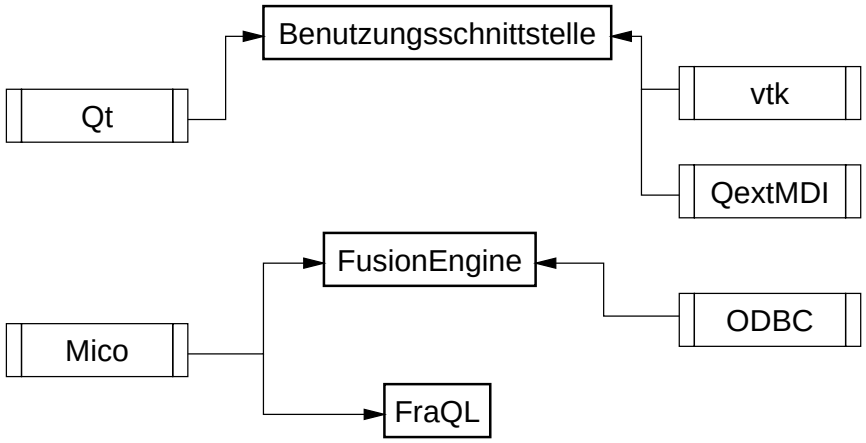


Abbildung 2.3: Verwendung von Bibliotheken

Hierbei werden nicht in allen Komponenten alle Bibliotheken verwendet. So fließt in FRAQL lediglich die MiCo-CORBA-Implementierung ein, die von allen Komponenten zur Verknüpfung miteinander verwendet wird. Selbstverständlich können die einzelnen FRAQL-Adapter ihrerseits weitere Bibliotheken einbinden. Da diese aber unabhängig vom FRAQL-System kompiliert und gelinkt werden, ist folglich zur Entwicklung keine weitere Bibliothek vonnöten.

Die Fusion Engine verwendet neben CORBA Teile von Qt und optional die ODBC-Bibliothek odbc++. Aus der Klassenbibliothek Qt werden in der Fusion Engine nur Klassen verwendet, die nicht der Visualisierung dienen. Dieses sind beispielsweise die Stringklasse QString oder Collections. Alle Klassen, die von QWidget abgeleitet sind, dürfen hier dagegen nicht verwendet werden. Die Bibliothek odbc++ ist eine frei verfügbare Implementierung zum Zugriff auf Datenbanken über ODBC-Treiber, die in ihrem API JDBC ähnelt. Sie kommt lediglich in der Fusion Engine zum Einsatz, wenn der Adapter für ODBC-Datenquellen in das FRAQL-System eingebunden wird. Ähnlich verhält es sich mit den nativen Bibliotheken von Datenbankherstellern, die je nach Bedarf an die entsprechenden Adapter gebunden werden.

In die Benutzungsschnittstelle fließen weitere Komponenten ein, die die Visualisierung ermöglichen oder vereinfachen. Neben den Visualisierungsklassen von Qt gehört dazu mittels QextMDI auch eine Erweiterung derselben um flexiblere MDI/SDI-Funktionalität. Visualisierungsaufgaben werden mittels OpenGL, Open Inventor sowie des Visualisierungstoolkits (VTK) umgesetzt. Die Anbindung an die Fusion Engine geschieht über CORBA. Somit müssen diese Bibliotheken zur Ausführung der FE nicht vorliegen.

2.1 Architektur der Fusion Engine

Die Fusion Engine stellt den zentralen Server-Prozess dar, der die Aktionen aller Anwender verwaltet und koordiniert. Client-Prozesse melden sich über eine CORBA-Schnittstelle an diesem an und können anschließend die zentral bereitgestellte Funktionalität nutzen. In Abbildung 2.4 sind der innere Aufbau sowie die Einbettung der Fusion Engine in die Gesamtumgebung detailliert dargestellt.

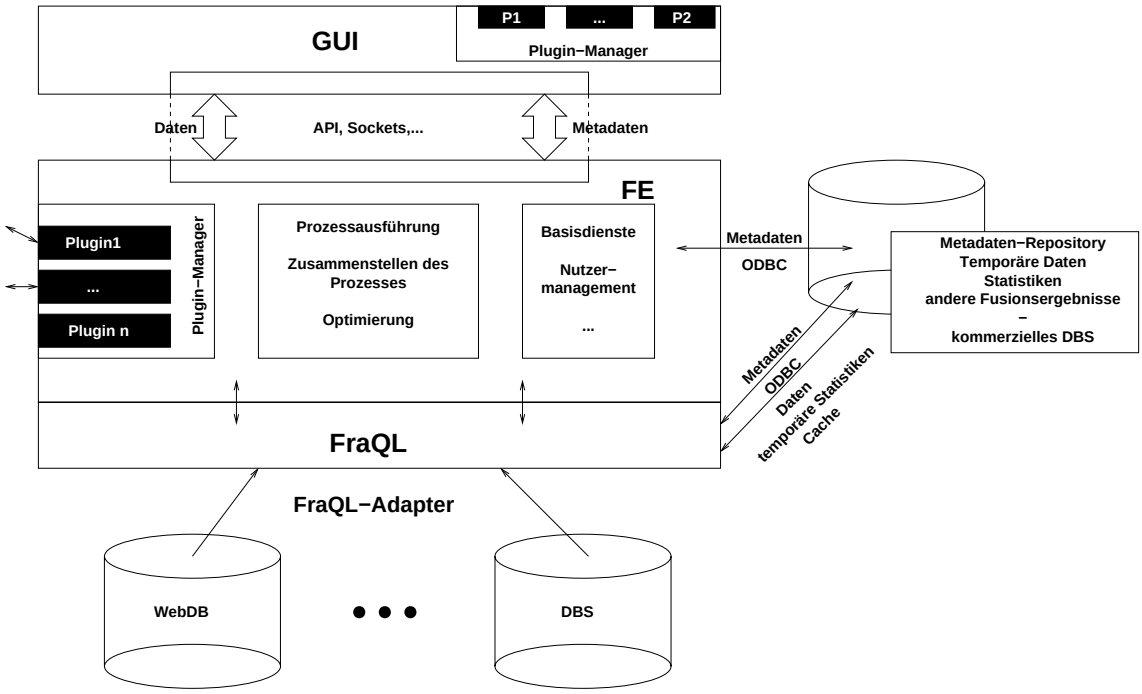


Abbildung 2.4: Architektur der Workbench

2.2 Architektur des Frontends ANGIE

Die Benutzungsschnittstelle ist gekennzeichnet durch ein Maximum am Flexibilität bezüglich der Auswahl von Darstellungs- und Interaktionstechniken für Informationsmengen, die während der einzelnen Phasen des Fusionsprozesses verfügbar sind. Dieses Ziel wird durch den komponentenbasierten Entwurf erreicht, wie er zum einen in Abbildung 2.4 im Rahmen der Gesamtsystemarchitektur und in Abbildung 2.5 als separater Ausschnitt gezeigt ist. Die einzelnen Plugins $P_1, P_2, \ldots P_n$ repräsentieren verschiedene Visualisierungskomponenten, welche dynamisch zur Laufzeit in das System geladen und wieder aus ihm entfernt werden können.

2.2.1 Das GUI

Innerhalb des GUIs findet eine Zweiteilung statt. ANGIE selbst bezeichnet lediglich die Klassen und Methoden zur Verwaltung der Oberfläche. Sie werden unterstützt durch die Bibliothek LIBANGIECORE, welche Funktionalitäten zur Verwaltung der Plugins sowie

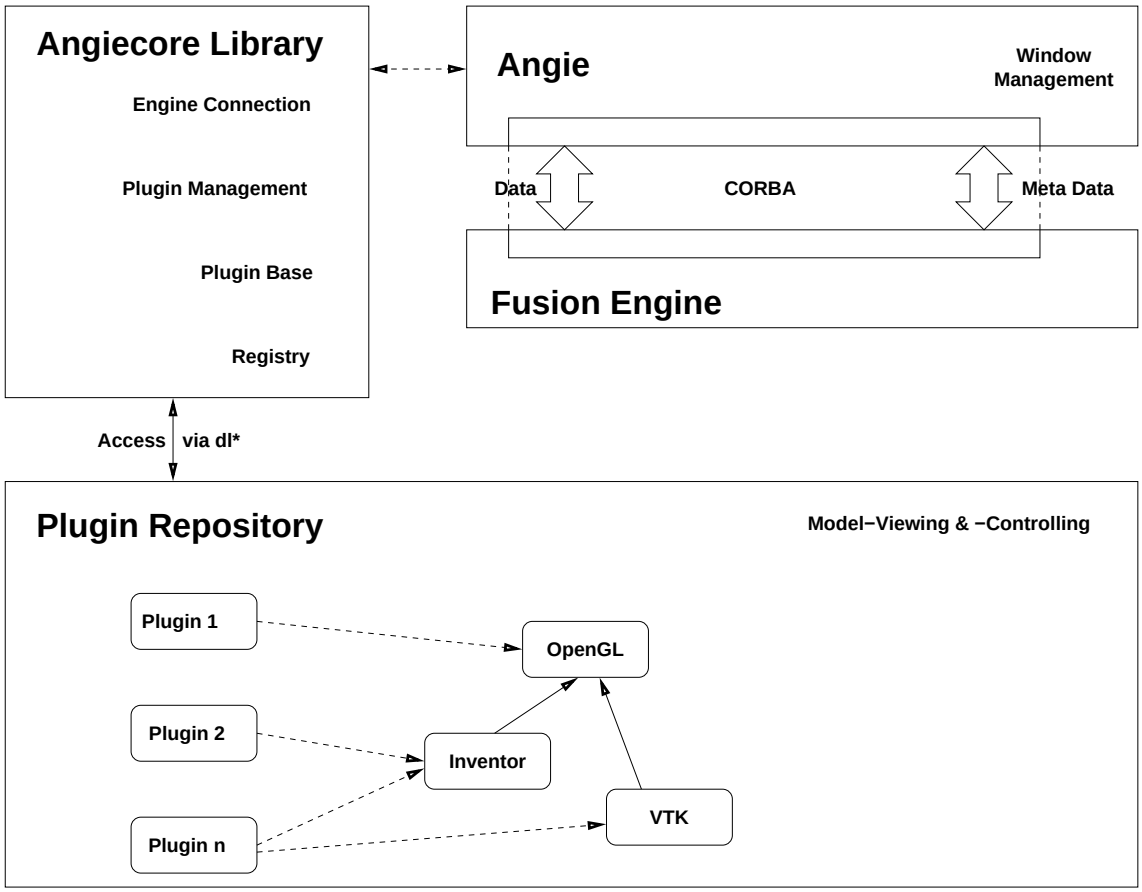


Abbildung 2.5: Architektur des Fusion-GUIs ANGIE

für die Verbindung von und zur Fusion Engine zur Verfügung stellt. Die Plugins dürfen ausschließlich Methoden, die von LIBANGIECORE bereit gestellt werden, verwenden. Ein direkter Zugriff auf ANGIE oder gar auf die Fusion Engine ist dagegen nicht erlaubt. Dadurch soll erreicht werden, dass die Rahmenbedingungen für die Entwicklung von Plugins minimal sind. Zusätzlich soll bei Weiterentwicklungen von ANGIE keine Neuübersetzung der Plugins notwendig werden.

2.2.2 Die Plugins

Jedes Plugin ergibt sich aus der Kombination von zumindest je einer Instanz einer Unterklasse von PluginBase sowie PluginWidget aus der CORE-Bibliothek dar. Abbildung 2.6 spiegelt dieses Prinzip wider. Die Aufteilung auf beide Klassen ist notwendig, da zum Einbinden in das Qt-basierte Frontend die Erzeugung eines Widgets notwendig ist. Dieses ist in einer separaten Klasse (PluginWidget) und nicht in PluginBase unterzubringen, da Objekte der Klasse QWidget in ihrem Konstruktor mit einem Eltern-Widget zu initialisieren sind. Ein nachträgliches Setzen dieses Eltern-Widgets ist zu einem späteren Zeitpunkt nicht möglich. Zum Zeitpunkt der Plugininitialisierung ist dieses Widget nicht notwendigerweise bekannt. Es reicht an dieser Stelle aus, die PluginBase-Unterklasse zu instanziieren und die Instanz des Widgets zu konstruieren, sobald alle zur Initialisierung dessen notwendigen Informationen verfügbar sind.

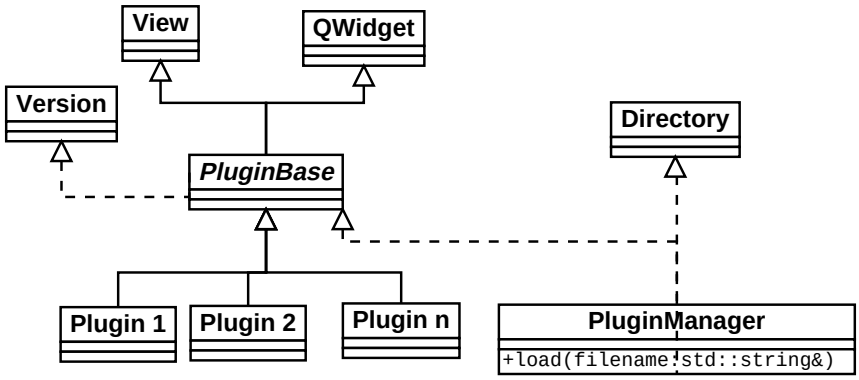


Abbildung 2.6: Klassenhierarchie von Angies Plugins

Installation

Jedes Plugin wird als eine “shared library” kompiliert und gelinkt. Eine beispielhafte tmake-Konfigurationsdatei ist folgendem Listing zu entnehmen.

```
TEMPLATE =      lib
CONFIG =        qt warn_on debug
INCLUDEPATH =   $(FUSIONDIR)/src/fuser
unix:TMAKE_LIBDIR +=    -L$(FUSIONDIR)/lib
unix:LIBS +=    -langiecore
HEADERS =       BeispielPlugin.h
SOURCES =       BeispielPlugin.cpp
TARGET =        BeispielPlugin
```

Die erzeugte Bibliothek wird in das Verzeichnis $FUSIONDIR/share/angie/ kopiert und steht somit zur Verwendung innerhalb des Fusions-Frontends ANGIE bereit. Sie kann bei Bedarf sofort in eine laufende Instanz nachgeladen werden.

3 Die Information Fusion Engine

3.1 Initialisierung

Bevor mit dem System gearbeitet werden kann, muss es initialisiert werden. Hierzu gehören u.a. der Aufbau der Datenbankverbindung(en) und das Nachladen von Komponenten (Operatoren). Der Ablauf des Initialisierungsprozesses ist in der Abbildung 3.1 dargestellt.

3.1.1 Herstellen einer Verbindung zur Metadatendatenbank

Mit der Anmeldung am System ist nicht (!) einfach die Anmeldung an der Datenbank mit den Metadaten gleichzusetzen. Hier sind im wesentlichen zwei Varianten denkbar: Entweder jeder Fusion Engine-Anwender entspricht tatsächlich einem Datenbankanwender, oder es existiert lediglich ein Datenbankanwender, und die Fusion Engine verwaltet Benutzerkonten vollständig selbstständig.

Wird für jeden Anwender der Fusion Engine ein Datenbankanwender in der Datenbank angelegt, hat dieses den Vorteil, dass viele der vorhandenen Fähigkeiten des DBMS in Bezug auf Benutzerverwaltung verwendet werden können. Der Nachteil ist bei dieser Variante die hohe Anzahl der Schemaobjekte, die die Wartung eines Systems mit vielen Anwendern erschweren können. Bei dieser Variante existiert ein Datenbankuser mit umfassenden Rechten, in dessen Schema die Metadatentabellen, die für alle Anwender gleich sind, angelegt werden. Die Rechte werden dann über Sichten den Fusion Engine-Anwendern zugeordnet. Zusätzlich sind diverse anwenderspezifische Relationen nötig, um persönliche Einstellungen ablegen zu können.

Bei der zweiten Variante gibt es nur ein einfaches DB-Schema, auf das ohne besondere Einschränkung der Rechte zugegriffen wird. Hier findet eine Verwaltung von Rechten lediglich durch Aktivierung und Deaktivierung von Funktionen durch das Anwendungsprogramm statt. Nachteile bei dieser Variante sind, dass bei direktem Zugriff auf die Datenbank mit den Metadaten alle Relationen für den Anwender beeinflussbar sind, und dass eine Datentrennung nicht möglich ist. Dagegen ist diese Variante einfacher zu implementieren.

Kann keine Verbindung zur Metadaten-Datenbank aufgebaut werden, ist ein Hochfahren des Systems (in der Produktionsversion) nicht möglich, da das System nicht korrekt initialisiert werden kann. Zur Erleichterung der Implementierung wird in der Entwicklungsversion bei Abbrechen der Anmeldung ein System mit “Dummy”-Metadaten hochgefahren. Dadurch kann auch hier die Entwicklung unabhängig von der Existenz einer Metadaten-Datenbank dezentral durchgeführt werden.

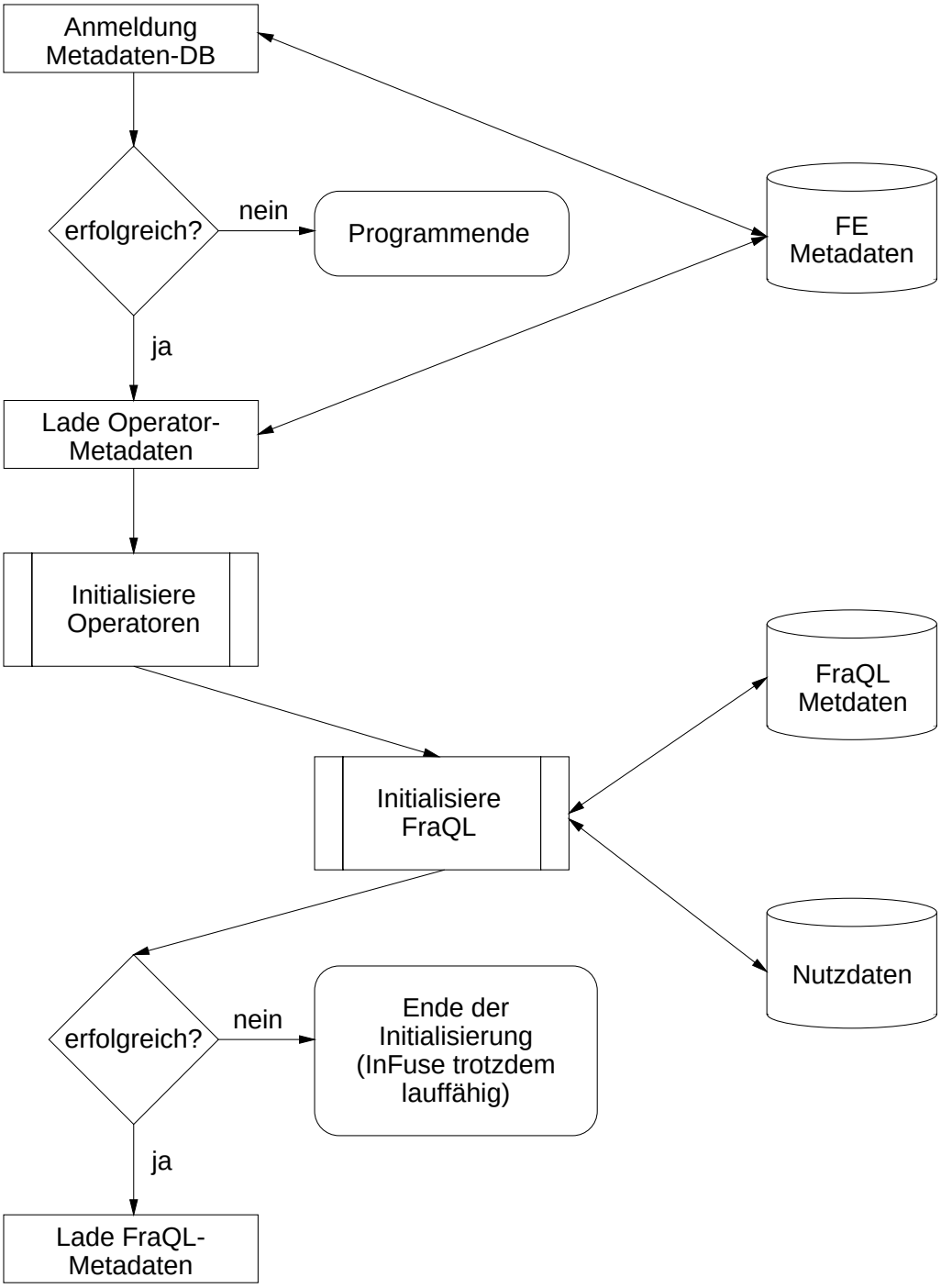


Abbildung 3.1: Flussdiagramm: Initialisierung der FE

3.1.2 Laden der Metadaten

Die zur Initialisierung notwendigen Metadaten werden geladen. Hierbei ist zu überlegen, ob zunächst nur die für den Programmstart notwendigen Daten geladen, und alle anderen on-demand nachgeladen werden. Zunächst wird auf Grund der einfacheren Implementierung die erste Variante implementiert.

3.1.3 Verbindung zu FRAQL

Da die Fusion Engine ohne FRAQL nicht auf Nutzdaten zugreifen kann (der direkte Zugriff über ODBC ist nur für die Metadaten vorgesehen) wird eine Verbindung zu FRAQL über CORBA aufgebaut. Scheitert die Herstellung einer Verbindung, wird dieses über den zentralen Fehlermechanismus propagiert. Ein Programmstart ist aber dennoch möglich, da auch ohne Zugriff auf die Nutzdaten sinnvolles Arbeiten möglich ist. So können administrative Tätigkeiten, für die lediglich das Vorhandensein der Metadaten-Datenbank nötig ist, durchgeführt werden.

3.2 Fehlerbehandlung

Eine zentrale Ausnahmebehandlung ist für die Verarbeitung aller Fehler zuständig. Dadurch wird erreicht, dass die Fusion Engine bei Auftreten ähnlicher Ereignisse in gleicher Weise reagiert, und dass nicht an mehreren Stellen mehrfach Fehlerbehandlungsroutinen implementiert werden müssen. Eine Weitergabe von nicht vollständig auflösbaren Fehlern an das Anwendungsprogramm ist vorgesehen.

3.2.1 Propagieren eines Fehlers

Eine zentrale Instanz (Singleton-Objekt) des Fehlerdienstes bietet Methoden, mit denen Ausnahmesituationen propagiert werden können. Damit diese in der Anwendungsebene signalisiert werden können, muss sich diese vorher beim Fehlerdienst als Empfänger registriert haben. Die weitere Verarbeitung signalisierter Fehler ist dann Aufgabe der Anwendung. In der Workbench können diese beispielsweise angezeigt werden, was in einer Batchanwendung selbstverständlich nicht möglich ist.

3.2.2 Protokollierung

Durch die zentrale Fehlerverwaltung ist eine Protokollierung derselben an zentraler Stelle möglich, um bei lang laufenden Prozessen eine Kontrolle zu ermöglichen. Hierzu sind auch verschiedene Fehlerkategorien wie Information, Warnung, Kritischer Fehler notwendig, um einen Abbruch von Prozessen bei unkritischen Fehlern zu vermeiden.

3.2.3 Interaktion

Die Fehlerinformation kann in geeigneten Fällen durch eine Information, wie der Fehler zu beheben ist, angereichert werden. Dadurch ist es der Anwendung beispielsweise

möglich, bei Fehlen eines Parameters automatisch den Dialog zur Eingabe desselben zu öffnen.

3.3 Fusionsoperatoren

3.3.1 Laufzeitumgebung

Die Fusionsoperatoren können als eine Form von in der Fusion Engine gespeicherten Stored Procedures betrachtet werden. Ihnen wird ein API zum Zugriff auf Daten bereitgestellt. Ein direkter Zugriff der Operatoren auf FRAQL unter Umgehung der Fusion Engine ist nicht vorgesehen, damit von ihnen die Basisdienste nicht umgangen werden können.

3.3.2 Dynamisches Laden

Die dem System bekannten Fusionsoperatoren (abgelegt in den Metadaten) werden während der Initialisierungsphase nachgeladen. Sollten Abhängigkeiten zwischen Operatoren erlaubt werden, müssen diese beachtet werden. In der ersten Implementierungsphase werden nur autarke Operatoren erlaubt, die unabhängig von der Existenz (und ggf. der Version) anderer Operatoren lauffähig sind. Ein geschachtelter Aufruf ist also zunächst nicht möglich, aber als Erweiterung für später vorgesehen. Wird ein in den Metadaten definierter Operator nicht gefunden oder kann dieser nicht geladen werden (beispielsweise weil die Bibliothek nicht vorhanden ist), werden die verbleibenden Operatoren dennoch geladen. Wird später ein vormals definierter Fusionsprozess geladen, muss also darauf geachtet werden, ob alle benötigten Operatoren betriebsbereit sind. Eine referentielle Integrität in den Metadaten reicht nicht aus.

Ein zweites Anwendungsfeld für das Nachladen ist die Administration von Operatoren. Ein Anwender mit Administrator- oder Entwicklerrolle kann neue Operatoren jederzeit in das System einbinden, so dass sie sofort zur Verfügung stehen.

3.3.3 Dynamisches Entladen

Ebenso, wie Operatoren geladen werden können, können sie auch, unter der Voraussetztung dass kein Fusionsprozess geöffnet oder gar aktiv ist, aus dem System entfernt werden. Hierdurch können durch nicht mehr benötigte Operatoren belegte Ressourcen freigegeben werden, oder neue Versionen eingebunden werden.

3.4 Fusionsprozesse

Bei den Fusionsprozessen muss zwischen der Definition und konkreten Instanzen unterschieden werden. Eine Definition ist dabei eine benannte Abfolge von Operatoren mit zugeordneten Datenquellen (und -senken). Ein Prozess ist eine parametrisierte und mit Statusinformationen angereicherte Instanz, die eine Definition als Template verwendet.

3.4.1 Prozess-Manager

Ein Singleton-Objekt verwaltet alle sich im Hauptspeicher der Engine befindlichen Prozessdefinitionen und Prozesse.

3.4.2 Definitionen

Eine Definition eines Fusionsprozesses ist eine azyklischer gerichteter Graph, dessen Knoten Operatoren und Datenquellen sind. Dieser Graph beschreibt die Aufeinanderfolge der am Prozess beteiligten Operatoren. Azyklisch ist der Graph zumindest in der ersten Ausbaustufe, da Iterationen und bedingte Abfolgen noch nicht unterstützt werden. In einer späteren Projektphase ist jedoch geplant, jede Kante des Graphen an eine Bedingung zu knüpfen, so dass Zyklen erlaubt werden können.

Neuerstellung

Ein neuer (leerer) Prozess wird über den Prozess-Manager (s.o.) mit einem Namen, also dessen Bezeichnung, angefordert. Wird der gewählte Name schon verwendet, wird diese Anforderung zurückgewsiesen, im anderen Fall wird eine neue Prozessdefinition mit dem Namen angelegt.

Abrufen

Da jede Prozessdefinition eindeutig durch ihren Namen identifiziert werden kann, kann eine bereits definierte Definition vom Prozess-Manager mit dem Namen als Schlüssel angefordert werden. Existiert der Name noch nicht, wird eine neue Definition erstellt und als Ergebnis geliefert.

Speichern

Die Definitionen werden unter ihrem Namen in der Metadaten-Datenbank gespeichert. Da diese parallel von mehreren Anwendern verwendet werden kann, ist hier ein geeignetes Transaktionskonzept vorzusehen. Es gibt zumindestens eine Option, beim Speichern einen neuen Namen anzugeben (Speichern unter), um Änderungen notfalls unter neuem Namen ablegen zu können.

Bearbeiten von Knoten

Als Knoten können Datenquellen und Operatoren zum Graph hinzugefügt werden. Hier werden auch Darstellungsoperationen mit verwaltet, damit sie in den Metadaten abgelegt werden können. Also speichert jeder Knoten neben dem Verweis auf den Operator oder die Datenquelle, die er repräsentiert, zumindest die Koordinaten bei graphischer Darstellung des Graphen und evtl. weitere Merkmale.

Werden Knoten aus dem Graph entfernt, werden alle assozierten Kanten ebenfalls gelöscht. Dadurch bleibt der Graph immer wohldefiniert.

Bearbeiten von Kanten

Die zeitliche Aufeinanderfolge und die Abhängigkeiten werden durch die Kanten des Graphen repräsentiert. Diese können eingefügt, geändert und gelöscht werden. Eine Einfügung ist nur zwischen bereits bestehenden Knoten möglich, um stets die Wohldefiniertheit zu erhalten (siehe auch Löschen von Knoten). Wird ein noch nicht existierendes Datenobjekt als Ziel einer Operation angegeben, so wird es erzeugt, bevor die entsprechende Kante eingefügt wird.

3.4.3 Ausführung

Nachdem ein Prozess definiert worden ist, kann er ausgeführt werden. Dazu wird eine Instanz des Prozesses erzeugt, initialisiert und gestartet. Der Ablauf des Prozesses soll später im Hintergrund erfolgen, so dass die Fusion Engine weiterhin interaktiv bleibt. Zunächst wird aber ein blockierender Aufruf umgesetzt. Auch muss der jederzeitige Abbruch einzelner Operatoren oder des gesamten Prozesses möglich sein, falls lang laufende Operationen gestartet worden sind, deren Ergebnis nicht mehr benötigt wird.

3.4.4 Erzeugen einer Prozess-Instanz

Die Prozess-Instanz wird durch den Prozess-Manager erzeugt, indem ein unparametrisierter Prozess, der zu einer namentlich benannten Definition assoziiert ist, erzeugt wird. Die Instanzen zu einer Definition werden fortlaufend nummeriert und auch als Worksheets bezeichnet.

3.4.5 Parametrisieren

Zu jedem Knoten des Prozessgraphs können Parameter angegeben werden. Die formale Parameterliste ist eine Eigenschaft des Operators. Die Liste mit den aktuellen Parameterwerten für jeden in der Definition verwendeten Operator dagegen ist eine Eigenschaft des Worksheets.

3.4.6 Persistenz

Ein Worksheet kann jederzeit gespeichert werden. Sollte die Definition noch nicht in der Metadaten-Datenbank abgelegt sein, wird diese zuerst gespeichert. Zu den Metadaten eines Prozesses gehören ein Verweis auf die Definition, die Parameter und Statusinformationen.

3.4.7 Starten

Nach einem Test auf Konsistenz, Vorhandensein der nötigen Parameter usw. wird die Reihenfolge der Operatoren bestimmt. Hierzu werden Quellen des Graphen bestimmt, die dann während und nach ihrer Ausführung entsprechend markiert werden, woraufhin neue Quellen bestimmt werden. Theoretisch können alle Quellen parallel gestartet werden, wenn die beteiligten Komponenten dieses unterstützen. Die Knoten des Prozesses können folgende Stati aufweisen:

- bereit,
- ausgeführt,
- wartend und
- fehlerhaft.

Nach Initialisierung des Prozesses bekommen alle Knoten den Status *wartend*. Anschließend werden die Quellen bestimmt und mit dem Status *bereit* versehen. Alle Knoten mit dem Status *bereit* werden ausgeführt. Je nach Erfolg der Ausführung bekommen sie den Status *ausgeführt* oder *fehlerhaft*. Iterativ werden alle nicht mit *fehlerhaft* markierten Quellen des Teilgraphen ohne Berücksichtigung der mit *ausgeführt* markierten Knoten auf den Status *bereit* gesetzt und ausgeführt. Dadurch wird erreicht, dass selbst bei Fehlern in einem Teilbaum alle erfolgreich abzuschliessenden Operationen durchgeführt werden. Die fehlerhaften Knoten können dann einzeln (beispielsweise nach Behebung von Fehlern in den Parametern) manuell wieder auf den Status *wartend* gesetzt werden, um sie bei der nächsten Iteration zu berücksichtigen.

3.5 Schnittstelle der Fusion Engine zur Anwendung

3.5.1 Beschreibung

Die Anwendung setzt auf der Information Fusion Engine auf und verwendet diese neben den Visualisierungs-Plugins als Komponente für den Datenzugriff. Dadurch wird einerseits eine Unabhängigkeit der Benutzungsschnittstelle von der eingesetzten Datenbanktechnologie erreicht, andererseits kann die Fusion Engine weitgehend unabhängig von der Bedienkomponente entwickelt werden.

Damit trotz der Trennung die Anwendungskomponente mit der Fusion Engine reibungslos zusammenarbeiten kann, bietet die Fusion Engine ein umfangreiches (CORBA-basiertes) API an, das Interaktion und Visualisierung unterstützt.

3.5.2 Authentifizierung

Anmeldung am System

Nach Übergabe der Anwenderkennung samt Passwort (und evtl. Connect-String) muss die Initialisierung des Systems mit diesen Parametern gestartet werden.

Abfragen von Rechten

Das Anwendungsprogramm muss in der Lage sein, das Vorhandensein von Berechtigungen (oder evtl. die Zugehörigkeit zu Rollen) abzufragen, um die Programmsteuerung entsprechend anzupassen (z.B. Ein-/Ausblenden von Menüs).

Administration von Anwenderberechtigungen

Ein Anwender, der in der Rolle des Administrators angemeldet ist, muss in der Lage sein, Benutzer anzulegen, zu ändern und zu löschen. Entsprechende Funktionalität ist von der Engine bereitzustellen.

3.5.3 Fusionsoperatoren

Liste geladener Fusionsoperatoren

Ein API-Aufruf muss eine Liste der zur Verfügung stehenden Fusionsoperatoren liefern, damit diese in der Anwendung angezeigt und zur Verwendung bereitgestellt werden können.

Laden und Entfernen

Das dynamische Hinzunehmen oder Entfernen von Fusionsoperatoren muss von der Anwendung aus initiiert werden können. Der Vorgang des Ladens und Entladens ist in Abschnitt 3.3 beschrieben.

Ausführen

Da nicht immer der gesamte Prozess gestartet werden kann, kann die Ausführung einzelner Operatoren gestartet werden.

Eigenschaften und Parameter abfragen

Für jeden der Fusionsoperatoren kann dessen Schnittstellenspezifikation abgefragt werden. Dazu gehören die Anzahl und Art der Ein- und Ausgangsrelationen sowie die Parameter.

3.5.4 Definition von Fusionsprozessen

Zum Erstellen und Bearbeiten von Prozessdefinitionen ist ein API notwendig, das Anlegen von Deklarationen, Ergänzen, Ändern und Löschen von Operatoren, Datenquellen und deren Verknüpfungen ermöglicht.

Speichern und Laden von Prozessdeklarationen und -prozessen

Sowohl Deklarationen von Prozessen, als auch die Worksheets (mit Parametern und Status) müssen gespeichert und geladen werden können.

Ausführen

Zum Ausführen eines Fusionsprozesses ist die Anreicherung einer Deklaration mit Parametern notwendig, die an die Fusion Engine übergeben werden. Voraussetzung hierfür ist, dass notwendige und optionale Parameter für jeden Knoten des Prozesses abgefragt werden können.

4 Zusammenfassung und Ausblick

Das vorgestellte Konzept ist in prototypischer Form bereits umgesetzt. Erfahrungen, die aus der Arbeit mit dem Prototypen resultieren sowie Anregungen und Inspirationen, die auf der CeBIT 2001 gesammelt wurden, bilden die Grundlage für die Ausarbeitung eines Feinkonzeptes. Dieses bildet die Grundlage für weitere Entwicklungsarbeiten. Die entstehende Fusionsworkbench bildet den Grundbaustein zur Validierung entworfener Konzepte sowie zur Evaluierung der Praxistauglichkeit entwickelter Methoden. Noch während der Entwicklungsarbeiten dient der Prototyp als Testumgebung für die Evaluierung verschiedener Visualisierungs- und Datamining-Bausteine.

Für eine weiterführende Phase des Projektes ist es geplant, den schon bestehenden iterativen Charakter der Software zu intensivieren. Mittels OLAP-Methoden werden Ergebnisse von Datenbankanfragen in Teilen schon während der Anfragebearbeitung der weiteren Auswertung zur Verfügung gestellt. Ausführliche Ergebnisse, die auch eine höhere Präzision aufweisen, erfordern entsprechend längere Bearbeitungs- und somit Wartezeiten für den Benutzer. Dadurch wird es möglich, dem Anwender ein grobes Bild der situativen Arbeitsschritte und ihrer Ergebnisse schnell zu präsentieren, so dass dieser entsprechend reagieren kann. Die Benutzerführung gewinnt an Qualität.

Eine weitere Arbeit für die Zukunft ist die formale Beschreibung des gesamten Prozesses der Informationsfusion von der Integration, Aufbearbeitung, Analyse bis hin zur Visualisierung. Diese Beschreibung könnte dabei einen anfrageorientierten Charakter besitzen, um den interaktiven und iterativen Charakter der Informationsfusion zu unterstützen. Weiterhin sind Anforderungen an die Metadaten für Datenquellen und Fusionsoperatoren sowie grafische Anzeigemöglichkeiten aufzustellen. Dabei ist eine Abstraktion ähnlich zu physischen und logischen Operatoren in der Datenbankwelt denkbar. Eine Analyse muss klären, ob diese Idee umsetzbar ist. Das Ziel einer solchen Spezifikation ist es einmal, den Anwender zu führen und andererseits, durch die Abstraktion eine automatische Optimierung des gesamten Prozesses der Informationsfusion zu erreichen.

Danksagung

Unser Dank gilt Dr. Kai-Uwe Sattler für die Unterstützung und Diskussion bei der Konzeption des Prototypen sowie für seine Implementation des FRAQL Anfragesystems, welches in dieser Arbeit verwendet wird.

Literaturverzeichnis

[CDH+99] John Clear, Debbie Dunn, Brad Harvey, Michael L. Heytens, Peter Lohman, Abhay Mehta, Mark Melton, Lars Rohrberg, Ashok Savasere, Robert M. Wehrmeister, and Melody Xu. NonStop SQL/MX Primitives for Knowledge Discovery. In *Proc. 5th ACM SIGKDD Int. Conference on Knowledge Discovery and Data Mining 1999, San Diego, CA USA*, pages 425–429, 1999.

[CFB99] Surajit Chaudhuri, Usama M. Fayyad, and Jeff Bernhardt. Scalable Classification over SQL Databases. In *Proceedings of the 15th International Conference on Data Engineering, 1999, Sydney, Austrialia*, pages 470–479. IEEE Computer Society, 1999.

[Cha98] Surajit Chaudhuri. Data Mining and Database Systems: Where is the Intersection? *Data Engineering Bulletin*, 21(1):4–8, 1998.

[CMN99] S. Chaudhuri, R. Motwani, and V.R. Narasayya. On Random Sampling over Joins. In A. Delis, C. Faloutsos, and S. Ghandeharizadeh, editors, *SIGMOD 1999, Proceedings ACM SIGMOD International Conference on Management of Data, June 1-3, 1999, Philadephia, Pennsylvania,USA*, pages 263–274. ACM Press, 1999.

[CSS99] S. Conrad, G. Saake, and K. Sattler. Informationsfusion - Herausforderungen an die Datenbanktechnologie. In A. P. Buchmann, editor, *Datenbanksysteme in Büro, Technik und Wissenschaft, BTW'99, GI-Fachtagung, Freiburg, März 1999*, Informatik aktuell, pages 307–316, Berlin, 1999. Springer-Verlag.

[DGJ+01] O. Dunemann, I. Geist, R. Jesse, G. Saake, and K. Sattler. INFUSE – Eine datenbankbasierte Plattform für die Informationsfusion. In *Datenbanksysteme in Büro, Technik und Wissenschaft (BTW 2001),* 2001. *to appear*.

[Eic00] Stephen G. Eick. Visualizing Multi-Dimensional Data. *Computer Graphics*, pages 61–67, February 2000.

[FPSSU96] U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors. *Advances in Knowledge Discovery and Data Mining*. AAAI Press / The MIT Press, Menlo Park, California, 1996.

[OR86] F. Olken and D. Rotem. Simple Random Sampling from Relational Databases. In W.W. Chu, G. Gardarin, S. Ohsuga, and Y. Kambayashi, editors, *VLDB'86 Twelfth International Conference on Very Large Data Bases,August 25-28, 1986, Kyoto, Japan, Proceedings*, pages 160–169. Morgan Kaufmann, 1986.

[Sch98] I. Schmitt. *Schemaintegration für den Entwurf Föderierter Datenbanken*, volume 43 of *Dissertationen zu Datenbanken und Informationssystemen*. infix-Verlag, Sankt Augustin, 1998.

[SCS00] K.-U. Sattler, S. Conrad, and G. Saake. Adding Conflict Resolution Features to a Query Language for Database Federations. *Australian Journal of Information Systems*, 8(1):116–125, 2000.

[SML98] Will Schroeder, Ken Martin, and Bill Lorensen. *The Visualization Toolkit – An Object-Oriented Approach to 3D Graphics*. Prentice Hall PTR, 2. edition, 1998.

[STA98] Sunita Sarawagi, Shiby Thomas, and Rakesh Agrawal. Integrating mining with relational database systems: Alternatives and implications. In Laura M. Haas and Ashutosh Tiwary, editors, *SIGMOD 1998, Proceedings ACM SIGMOD International Conference on Management of Data, 1998, Seattle, Washington, USA*, pages 343–354, 1998.

[Vit87] J.S. Vitter. An Efficient Algorithm for Sequential Random Sampling. *ACM Transactions on Mathematical Software*, 13(1):58–67, March 1987.