University of Zurich
Department of Informatics

Export von Datenbankinhalten in Datenformate von Statistikprogrammen

Bachelor Thesis in Informatics

submitted by

Stefan Schurgast

Rapperswil-Jona, Switzerland

Student ID: 02-913-770

Supervisor: Boris Glavic

Prof. Klaus R. Dittrich & Prof. Abraham Bernstein

Date of Submission: December 30th, 2007

University of Zurich

Department of Informatics (IFI)

Database Technology Research Group

Binzmühlestrasse 14, CH-8050 Zürich, Switzerland

I Zusammenfassung

Das sesamDB Projekt ist ein Teilprojekt der interdisziplinären Langzeitstudie sesam zur Ätiologie von psychischen Erkrankungen. Es beschäftigt sich mit der Entwicklung der Datenbank für wissenschaftliche und administrative Daten von sesam sowie der Implementierung verschiedener Clientanwendungen. Um die in sesam erhobenen Daten mittels Statistiksoftware analysieren zu können, wurde eine Applikation entwickelt, die Daten aus der sesamDB in gängige Statistikformate exportiert. Eine grafische Benutzeroberfläche ermöglicht es dem Anwender, die benötigten Daten ohne Kenntnisse über den Datenbankaufbau oder Datenanfragesprachen zu erhalten. Diese Arbeit enthält eine Zusammenstellung verwandter Arbeiten sowie den Entwicklungsprozess und die Architektur des Exportprogramms, Sesam Export Manager.

II Abstract

The sesamDB Project is a subproject of the interdisciplinary long time study sesam about the etiology of mental health. Its main task is to develop a database for scientific and administrative data for sesam as well as the implementation of client applications. In order to analyze the stored data with statistical analysis software, Sesam Export Manager has been built to extract data from sesamDB to data types of popular statistics applications. The therefore developed graphical user interface helps the user to obtain the data he needs without having knowledge of the underlying database schemes or query languages. This paper contains a composition of related work, the development process and the architecture of Sesam Export Manager.

III Anmerkung

Diese Bachelorarbeit wurde durch die Database Technology Research Group des Instituts für Informatik der Universität Zürich unterstützt. Sämtliche in dieser Arbeit ausgedrückten Meinungen, Ergebnisse und Schlussfolgerungen sind diejenigen des Autors und entsprechen nicht notwendigerweise der Sichtweise der vorher genannten Parteien.

An dieser Stelle möchte ich mich insbesondere bei Boris Glavic herzlich für die gute Unterstützung während der gesamten Ausarbeitung dieser Bachelorarbeit bedanken.

IV Inhaltsverzeichnis

V Abbildungsverzeichnis

VI Abkürzungsverzeichnis

ACID	Atomarität, Konsistenz, Isolierheit und Dauerhaftigkeit
API	Application Programming Interface
AWT	Abstract Window Toolkit
CIDI	Clinical Interview Diagnosis Instrument
CSV	Comma Separated Values
DBI	Database Independent
EJB	Enterprise Java Beans
GUI	graphical user interface
HQL	Hibernate Query Language
JAR	Java Archive
JDBC	Java Database Connectivity
JDO	Java Data Objects
JULP	Java Ultra-Lite Persistence
MVC	Model-View-Controller
NFS	Nationale Forschungsschwerpunkte
ODBC	Open Database Connectivity
ORDBMS	objektrelationales Datenbanksystem
ORM	Object-Relational Mapping
SAS	Statistical Analysis Systems
sesam	Swiss Etiological Study of Adjustment and Mental Health
SNF	Schweizerischer Nationalfonds
SPSS	Statistical Product and Service Solution
SWT	Standard Widget Toolkit
WHO	Weltgesundheitsorganisation

1 Einleitung

1.1 Ausgangslage und Problemstellung

Die Verwaltung von erhobenen Daten aus Langzeitstudien stellt die Studienverantwortlichen immer wieder vor Schwierigkeiten. Zum einen müssen grosse Datenmengen so gespeichert und wieder aufgerufen werden können, dass sie den genauen Sachverhalt wiedergeben. Zum anderen müssen Bestimmungen zum Datenschutz eingehalten werden, um die Probanden in ihrer Person zu schützen (sesam Schweiz, 2007).

Sesam, die „Swiss Etiological Study of Adjustment and Mental Health“, ist eine psychologische Langzeitstudie (siehe Kapitel 2: Das sesam Projekt) und ist Teil des Nationalen Forschungsschwerpunkts des Schweizerischen Nationalfonds. Mit der sesamDB, einer Teilstudie von sesam, steht ihr eine für sie zugeschnittene Datenbank für die Verwaltung von Untersuchungs- und Administrationsdaten zur Verfügung, die oben erwähnten Anforderungen gerecht wird (sesam Schweiz, 2007).
Sobald die Daten jedoch zur Auswertung in einem Statistikprogramm benötigt werden, gestaltet sich der Zugriff auf die Datenbank als schwierig. Die sesamDB unterstützt lediglich einen Zugriff mittels der Datenbankanfragesprache SQL. Es kann jedoch nicht davon ausgegangen werden, dass jeder, der Daten zur Weiterverarbeitung benötigt, auch SQL beherrscht oder den Aufbau der Datenbank kennt. Ausserdem ist für jeden Verwendungszweck eine andere Sicht auf die zugrundeliegenden Daten notwendig. Hinzu kommt, dass gewisse Informationen wie personenspezifische Daten oder Daten über den Aufbau der Datenbank dem Benutzer bewusst vollständig vorenthalten werden sollen.

1.2 Zielsetzung und Vorgehensweise

Ziel dieser Arbeit ist es, eine Clientanwendung (im Folgenden auch Sesam Export Manager genannt) zu erstellen. Mit dieser soll sich der Benutzer von Studiendaten bequem und über eine grafische Benutzeroberfläche und mithilfe der darunterliegenden Exportlogik die benötigten Datenexporte im richtigen Datenformat für das von ihm für die Weiterverarbeitung der Daten verwendete Statistikprogramm zusammenstellen können. Die Applikation soll genauso für Benutzer mit wenig technischen Kenntnissen im Bereich Datenbankanfragen wie auch für den Spezialisten auf einfache und leicht verständliche Weise verwendbar sein. Aus Gründen des Datenschutzes dürfen keine personenspezifischen Daten zur Auswertung vorhanden sein. Deshalb müssen die Daten während des Exports pseudonymisiert werden.
Als Ergebnis dieser Arbeit steht dem Anwender somit eine vollumfänglich funktionstüchtige und dokumentierte sowie an einer beispielhaften Ausprägung der Datenbank getestete Exportapplikation zur Verfügung. Neben dieser wird eine Installations- als auch eine Konfigurationsanleitung erstellt, die es dem Administrator erleichtert, die fertige Software zuverlässig und einfach in Betrieb nehmen zu können.

Das Vorgehen für die Erstellung des Sesam Export Managers ist wie folgt: In einer ersten Phase werden, wie die unten illustrierte Abbildung 1 zeigt, alle für die Applikation notwendigen Anforderungen spezifiziert und nach Prioritäten sortiert. Danach erfolgt die Planungsphase der Software, in welcher die Architektur festgelegt und anschliessend ein Detailentwurf erstellt wird. Gleichzeitig werden die zur Implementierung benötigten Hilfsmittel ausgesucht und definiert. Nach der konzeptionellen Phase wird die Applikation in einem dritten Schritt implementiert und auf ihre Funktionalität getestet. Abschliessend wird die fertige Lösung evaluiert.

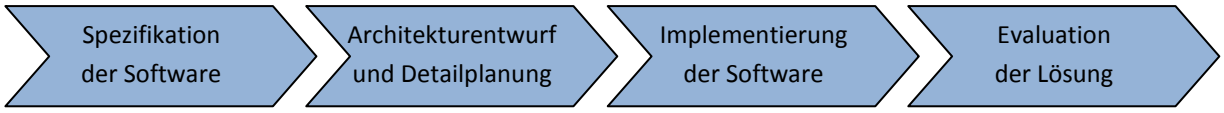

Abbildung 1: Entwicklungsprozess des Sesam Export Managers

1.3 Aufbau der Arbeit

Inhaltlich besteht die vorliegende Arbeit, wie die nachfolgende Abbildung 2 zeigt, aus 7 Teilen. Nach der Einleitung in Kapitel 1 wird in Kapitel 2 das Sesam-Projekt vorgestellt. Anschliessend folgen in Kapitel 3 verwandte Arbeiten zur Thematik. Die ersten drei Kapitel stellen somit den äusseren Rahmen dieser Arbeit dar und dienen dem besseren Verständnis für die Arbeit am Sesam Export Manager. Der Übersichtlichkeit halber sind sie deshalb in der unten eingefügten Abbildung grau eingefärbt.
In Kapitel 4 werden Anforderungen an eine Exportanwendung für die Daten der sesam Datenbank analysiert. Diese ermöglichen Aussagen über einen geeigneten Aufbau der zu erstellenden Applikation. In Kapitel 5 wird unter 5.1 erst der konzeptionelle Aufbau der Anwendung, unter 0 anschliessend ihr Detailentwurf erarbeitet. Zur vereinfachten Implementierung werden Hilfsmittel wie Toolkits und Frameworks eingesetzt, welche in Kapitel 5.3 ausgesucht werden. Die während der Umsetzung der Applikation aufgetretenen Probleme werden in Kapitel 5.4 analysiert und geeignete Lösungsansätze vorgestellt. Die fertige Exportanwendung wird daraufhin in Kapitel 6 bezüglich ihres Aufbaus sowie ihrer Funktionalität evaluiert. Abschliessend folgen in Kapitel 7 ein Fazit über die durch die Arbeit gewonnenen Resultate und ein Ausblick auf mögliche weiterführende Arbeiten. Da die eben erwähnten Kapitel 4 bis 7 die Ergebnisse und Erkenntnisse dieser Arbeit aufzeigen, werden sie in der folgenden Abbildung 2 blau dargestellt.

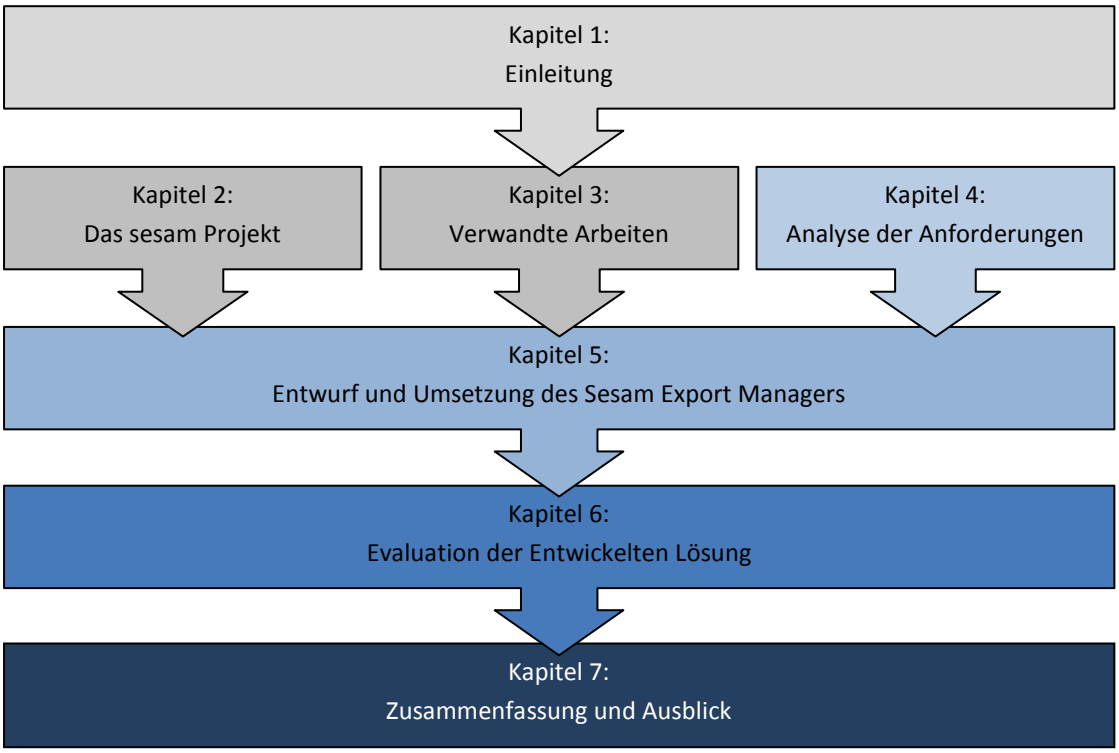

Abbildung 2: Aufbau der Arbeit

2 Das sesam Projekt

Als Grundlage für die folgenden Kapitel wird an dieser Stelle das sesam Projekt vorgestellt. Dabei wird zuerst im Kapitel 2.1 auf die Kernstudie eingegangen. Anschliessend folgt in Kapitel 2.2 eine Einführung ins Teilprojekt N, das Projekt sesamDB, zu welchem auch die vorliegende Arbeit und die erstellte Exportanwendung für Statistikdaten, der sesam Export Manager, angehört.
Kapitel 2.3 zeigt auf, aus welchen Gründen sich das sesamDB Projektteam der Universität Zürich für die Haltung der Studiendaten für eine Datenbanklösung entschieden hat. In Kapitel 2.4 wird dargestellt, mit welcher Technologie die Datenbank realisiert wurde und Kapitel 2.5 beschreibt, wie die Datenbank im sesam Projekt konkret aufgebaut ist.

2.1 Einführung in das sesam Projekt

Einer Studie der Weltgesundheitsorganisation [WHO] zur Folge sind in den Industriestaaten psychische Erkrankungen, insbesondere Depressionen, zurzeit die viert häufigste Ursache von schwerwiegenden gesundheitlichen Beeinträchtigungen oder sogar vorzeitiger Sterblichkeit. Die Tendenz zu Depressionserkrankungen ist jedoch stark steigend, weshalb man bereits für das Jahr 2020 damit rechnet, dass Depressionen Verkehrsunfälle als zweit häufigste Ursache verlorener Lebensjahre durch schwerer gesundheitlicher Beeinträchtigung oder vorzeitiger Sterblichkeit ablösen werden (Murray & Lopez, 1997). Derzeit liegt die durchschnittliche Wahrscheinlichkeit einer psychischen Erkrankung bei rund 40%, die Wahrscheinlichkeit im Laufe des Lebens an einer Depression zu erkranken bei 15% (Gaebel, 2007). Diese Entwicklung ist beängstigend und gibt sesam Anlass, dem entgegenzuwirken.

Sesam ist gemäss sesam Schweiz eine interdisziplinäre Langzeitstudie der Universität Basel in Zusammenarbeit mit mehreren Partnerinstitutionen[1] und Spitälern in der ganzen Schweiz[2]. Die Abkürzung steht für „Swiss Etiological Study of Adjustment and Mental Health", zu Deutsch "Schweizerische ätiologische Studie zur psychischen Gesundheit". Diese Langzeitstudie begleitet 3000 Kinder ab der 20. Schwangerschaftswoche zusammen mit ihren Familien (Eltern und Grosseltern) über 20 Jahre bis ins junge Erwachsenenalter. Dabei werden in der Kernstudie psychologische, genetische, umweltbedingte und soziale Faktoren betrachtet (2007).

Das Ziel des Projekts besteht einerseits darin, die komplexen Ursachen, die zu einer gesunden psychischen Entwicklung führen, aufzudecken, als auch Ursachen und Auslöser für psychische Störungen frühzeitig zu erkennen. Diese gewonnenen Erkenntnisse sollen helfen, wirksame Prävention sowie Behandlung und Bewältigungsstrategien bei psychischen Krankheiten und Lebenskrisen zu entwickeln (sesam Schweiz, 2007).

[1] Zu den Partnerinstitutionen zählen: Universität Zürich, Universität Düsseldorf (D), University of Warwick (GB), Nationaler Forschungsschwerpunkt
[2] Partnerspitäler sind: Frauenklinik Universitätsspital Basel, Frauenklinik Universitätsspital Zürich, Frauenklinik Inselspital Bern, Maternité Hôpitaux Universitaires Genève, Maternité CHUV Lausanne

Zusätzlich zur eben beschriebenen Kernstudie (A) besteht sesam aus weiteren 13 Teilstudien (sesam Schweiz, 2007):

Teilstudie B: Risiko Reduktion
Teilstudie C: Befinden in der Schwangerschaft
Teilstudie D: Bedeutung der Grosseltern
Teilstudie E: Schwangerschaft und psychische Störungen
Teilstudie F: Neurotizismus
Teilstudie G: Genetik
Teilstudie H: Modell elterlicher Vernachlässigung bei nicht-menschlichen Primaten
Teilstudie I: Blinzelreaktion
Teilstudie J: Autobiographie
Teilstudie K: Soziale Determinanten
Teilstudie L: Familienprozesse
Teilstudie M: Autonomes Nervensystem
Teilstudie N: Entwicklung einer Datenbank (siehe sesamDB in Kapitel 2.2)

Unterstützt wird das sesam Projekt durch den Nationalen Forschungsschwerpunkt (NFS), einem Förderungsinstrument des Schweizerischen Nationalfonds (SFN) (Schweizerischer Nationalfonds, 2005).

2.2 Teilprojekt N: sesamDB

Sesam als interdisziplinäre Langzeitstudie über einen Zeitraum von 20 Jahren erzeugt eine grosse Menge an Daten, darunter Fragebögen, biologische Analysen, genetische Daten, Multimediainhalte und Sequenzdaten, die über den Zeitraum der Studie hinaus in ihrer Form wie auch in ihrer Semantik erhalten bleiben soll (Glavic & Dittrich, 2006, S. 2). Persönliche Daten über die Studienteilnehmer unterliegen ausserdem dem Datenschutz und müssen deshalb vor Datenverlust und unberechtigtem Zugriff geschützt werden. Um einem Verlust der Datenqualität durch uneinheitliche Speicherung, aber auch Sicherheitsmängel vorzubeugen, ist ein einheitliches Datenmanagement erforderlich. Ausserdem sollen für alle Formen des Datenzugriffs Client-Anwendungen erstellt werden, die so den komplexen und damit zeitaufwendigen Zugriff auf die Daten vereinfacht.
Das Ziel von sesamDB ist der Entwurf und die Implementierung einer Datenbank inklusive der dazugehörigen Anwendersoftware, um die durch die Studie gesammelten wie auch administrativen Daten optimal zu verwalten und später auswerten zu können. Ausserdem sollen die eigens für sesam entwickelten Anwendungen helfen, die Abläufe während der Studie zu vereinfachen oder gar ganz zu automatisieren. Der Schutz der Daten wird durch ein Sicherheitskonzept sichergestellt.

Der Nutzen des Teilprojekts sesamDB ist also eine starke administrative Vereinfachung der Datenhaltung sowie eine bedeutende Verbesserung der Datenqualität und Sicherheit für das Gesamtprojekt.

2.3 Datenmanagement in Langzeitstudien

In Langzeitstudien wird dem Datenmanagement erfahrungsgemäss ungenügend Beachtung geschenkt. Oft werden nur Personendaten in Datenbanken abgelegt, die übrigen Daten werden direkt in den betreffenden Statistikprogrammen wie beispielsweisse SPSS zur Auswertung gespeichert. Das führt üblicherweise nach einiger Zeit zu chaotischen Datenbeständen, aus denen nur mit grossem zeitlichem Aufwand für das Zusammenstellen oder das Korrigieren von Daten, Auswertungen vorgenommen werden können. Auch die Datenqualität leidet stark unter einer unorganisierten Datenablage. Gründe dafür und wie dem mit einer allen Aspekten umfassenden Datenbank entgegengewirkt werden kann, sind in den folgenden Abschnitten ersichtlich.

2.3.1 Grosse Datenmengen

Eine Studie mit einer Laufzeit von 20 Jahren generiert riesige Datenmengen, bestehend aus wissenschaftlichen und administrativen Daten und den dazugehörigen Metadaten. Dies lässt die Frage aufkommen, wie diese gespeichert werden sollen. Würden alle Daten einfach in einzelne Dateien mit Formaten wie Excel-Arbeitsmappen abgelegt, hätte das zur Folge, dass irgendwann die maximale Grösse einer Datei erreicht wäre (in Excel z.B. die maximale Anzahl Zeilen). Die Daten müssten also auf mehrere Dateien aufgeteilt werden. Dies würde es wiederum stark erschweren, bei der Anfrage und Analyse einen einheitlichen Blick auf die Gesamtheit der Daten zu bekommen. Ausserdem könnten grosse Datenmengen das System vor Performanceprobleme stellen.

→ Durch die Verwendung einer Datenbank können die grossen Datenmengen am selben Ort einheitlich gespeichert werden, was die Datenhaltung enorm erleichtert. Sie ermöglicht die Verwendung eines einzigen Speicherorts für alle Daten und einheitliche Datenformate. Die Menge der Daten in einer PostgreSQL-Datenbank beispielsweise ist nur begrenzt durch den Speicher, der ihr zur Verfügung steht. Ausserdem kann eine Datenbank wie PostgreSQL (siehe dafür auch Kapitel 16**Fehler! Verweisquelle konnte nicht gefunden werden.**) Daten schnell und effizient laden. Auch Mehrfachzuordnungen, die zum Beispiel für mehrsprachige Beschreibungen benötigt werden, sind so ohne doppelte Datenhaltung möglich.
Zur Auswertung und Analyse wäre es sinnvoll, nur die erforderlichen Daten in der benötigten Granularität oder Aggregation auswählen zu können. Dies würde die Handhabung beispielsweise im Statistikprogramm erleichtern und Rechenzeiten kürzen.

2.3.2 Langer Zeitraum

Werden Daten über einen langen Zeitraum gesammelt und gesichert, müssen sie auch Jahre später noch zur Verfügung stehen. Ändern sich die IT-Systeme und werden die verwendeten Dateiformate dann nicht mehr unterstützt, so kann dies zu einem Problem werden.

→ Werden die Daten durch ein verbreitetes Datenbankmanagementsystem verwaltet, so werden auch Jahre danach noch Zugriffsmöglichkeiten auf die Daten bestehen. Ist der Quellcode des Datenbanksystems offengelegt, so könnten bei Bedarf Erweiterungen für das Datenbanksystem implementiert oder Anpassungen am System vorgenommen werden.

Auch für die Exportanwendungen könnte der lange Zeitraum zum Problem werden. Erstens ist es möglich, dass diese irgendwann nicht mehr auf neuen Betriebssystemen lauffähig sind. Die Exportanwendungen müssten dann auf eine neue Plattform portiert werden. Wahrscheinlicher aber ist der zweite Fall, dass das für den Export verwendete Statistikformat nicht mehr mit aktuellen Statistikprogrammen kompatibel ist. Bei Letzterem würde eine Anpassung des Exportskripts ausreichen. Die Verwendung von leicht erneuerbaren Export-Plugins würde sich deshalb anbieten.

Neben den technischen Herausforderungen zur Speicherung von Daten aus der Langzeitstudie, gibt es auch inhaltliche Schwierigkeiten, die sich während der Entwicklung stellen. So ändern mit Sicherheit viele Stammdaten (wie beispielweise der Name und die Wohnadresse des Studienteilnehmers oder es entstehen neue Elternverhältnisse für die untersuchten Kinder).

→ Um dies repräsentieren zu können, ist es nötig, Gültigkeitszeiträume von Daten zu definieren. In sesamDB ist dies mittels des temporalen Datenbankansatzes umgesetzt worden. Dies bedeutet, dass alle Daten inklusive ihrer Änderungen und ihrer Gültigkeit in der Datenbank gespeichert werden und somit jeder Stand der Datenbank wieder hergestellt und abgefragt werden kann.

2.3.3 fehlende Transaktionssicherheit

Neben organisatorischen Gründen, die für eine einheitliche Datenhaltung in einer Datenbank sprechen, kommen auch Qualitätsanforderungen an die Transaktionen hinzu. In der Datenbanktheorie sind diese unter der Kurzform ACID-Eigenschaften bekannt, wobei ACID ein Akronym für Atomarität, Konsistenz, Isoliertheit und Dauerhaftigkeit ist. Sie beschreiben wünschenswerte, aber oftmals ungenügend berücksichtigte Eigenschaften von Transaktionen (Elmasri & Navathe, 2005, S. 423). Im Folgenden werden diese einzeln kurz erklärt und die Auswirkungen auf die vielfach unorganisierte Datenhaltung in Langzeitstudien betrachtet.

Atomarität

Eine atomare Operation ist eine Operation, die entweder ganz oder gar nicht ausgeführt wird (Elmasri & Navathe, 2005, S. 423). Datenbankmanagementsysteme verhalten sich so, als ob eine elementare Operation ausgeführt würde, die nicht durch andere Operationen unterbrochen werden könnte. Dies ist wichtig, da die Daten zur korrekten Darstellung eines Sachverhalts vollständig sein müssen.
Eine manuelle Eingabe in einem Statistikprogramm ermöglicht im Normalfall keine atomare Operation, da die auszuführenden Operationen oft aus mehreren Einzeloperationen bestehen.

Konsistenz

Nach Durchführung der Transaktion müssen wieder die inhärenten und explizit definierten Integritätsbedingungen gelten, die auch schon vor der Durchführung der Transaktion feststanden. Dies gilt insbesondere für Schlüssel- und Fremdschlüsselbedingungen (Elmasri & Navathe, 2005, S. 423).
Durch eine Sicherung von Daten in verschiedenen Systemen findet jedoch keine Konsistenzprüfung statt. Die Daten müssen für übergreifende Auswertungen mühsam zusammengeführt werden. Insbesondere die Pflege von Stammdaten kann zum Problempunkt solcher verteilten, unorganisierten Datensysteme werden.

Isolation

Das Prinzip der Isolation bedeutet die Trennung von Transaktionen, sodass laufende Transaktionen sich nicht gegenseitig beeinflussen (Elmasri & Navathe, 2005, S. 423).
Werden zur Sicherung der gewonnenen Daten unterschiedliche Systeme ohne oder mit nur eingeschränkter Mehrbenutzerfunktonalität verwendet, so kann auch die Isolation einer Transaktion nicht garantiert werden, da sich die verschiedenen Transaktionen gegenseitig beeinflussen können.

Dauerhaftigkeit

Eine Transaktion ist dann dauerhaft, wenn nach einer erfolgreich abgeschlossenen Transaktion die Wirkung beständig bleibt. Sie dürfen nicht aufgrund von Fehlern verloren gehen (Elmasri & Navathe, 2005, S. 423). Auch nach Systemabstürzen müssen die Daten zur korrekten Wiedergabe eines Sachverhalts vollständig im System enthalten sein.
Unorganisierten Datensystemen fehlen die dafür nötigen Unterstützungsprozesse. Gängige Datenbanksysteme bieten solche Funktionalität meist an.

Die soeben aufgezählten Anforderungen an das Datenmanagement in einer Langzeitstudie machen deutlich, dass nur ein übergreifendes Konzept unter Verwendung eines Datenbanksystems alle Bedürfnisse erfüllen kann.

2.4 Datenbankmanagementsystem PostgreSQL

Bei PostgreSQL handelt es sich nach Angaben der PostgreSQL Global Development Group um eines der ältesten und am weitesten fortgeschrittenen objektrelationalen Datenbanksysteme [ORDBMS]. Sie unterstützt neben einer Reihe von eigenen Erweiterungen den SQL92 und den SQL 99 Standard. PostgreSQL ist eine freie Datenbank unter der BSD-Lizenz, womit ihr Quellcode für Erweiterungen oder Verbesserungen offen steht. Sie gilt unter den lizenzkostenfreien Datenbanken als eine der stabilsten und zuverlässigsten überhaupt, ist in ihrer Grösse lediglich durch den zur Verfügung stehenden Speicher beschränkt und bietet eine hohe Transaktionssicherheit an. PostgreSQL bietet neben Regelsystemen (Rules), Sichten (Views) und Triggern auch die Möglichkeit, Typen zu bilden und zu vererben oder gespeicherte Prozeduren (stored Procedures) zu nutzen (2007).

Ausserdem ist sie plattformunabhängig[3] und lässt sich deshalb leicht auf verschiedenen Umgebungen verwenden. PostgreSQL besitzt eine lebendige Community aus Entwicklern und Anwendern, wodurch die Weiterentwicklung gesichert ist. Aber auch Fragen bei Problemen können so rasch geklärt werden. Zur erleichterten Verwendung existieren zudem grafische Administrations- und Datenbankdesigntools wie pgAdmin, pgAccess und Tora.

[3] Die letzte stabile Version unterstützt 34 Betriebssysteme, darunter verschiedene Unix-Varianten und Windows (PostgreSQL Global Development Group, 2007).

2.5 Sesam Datenbank

Die sesam-Datenbank (sesamDB) ist eine auf PostgreSQL aufgesetzte Datenbank zur Speicherung aller administrativen und wissenschaftlichen Daten des sesam Projekts[4]. Die Datenbankschemas wurden von den Entwicklern von sesamDB in folgende sechs Kategorien aufgeteilt, welche je ein Teilschema bilden:

- *Kategorie **Rot**: Prozesse, Experimente und Messgrössen*
  In der roten Kategorie werden die Zusammenhänge zwischen den für die Studie erhobenen Daten, den Experimenten und Prozessen beschrieben. Anhand dieser Metadaten werden später im Sesam Export Manager die für den Export von Datenobjekte nötigen ausgewählt.

- *Kategorie **Grün**: wissenschaftliche Daten und Instrumente*
  Dieser Teil enthält alle wissenschaftlichen Erhebungsdaten. Aufgrund der unterschiedlichen Eigenschaften von Experimenten können diese Daten von einfachen Integerwerten (wie beispielsweise ein Barcode auf einer Blutprobe) bis hin zu komplexen Inhalten (wie vollständigen Fragebogen) reichen.

- *Kategorie **Blau**: Equipment und Speicherung*
  In diesem Schema werden alle zur Erfassung und Verwaltung der Studiendaten relevanten Messgeräte, Behältnisse und andere Materialien sowie Standorte festgehalten.

- *Kategorie **Braun**: Personen, Probanden und Angestellte*
  Die Kategorie Braun enthält das Teilschema über alle an der Studie involvierten Personen wie Probanden und Studienteilnehmer, deren Familien und Angestellte.

- *Kategorie **Violett**: Orte, Studien und Organisationen*
  Dieses Teilschema beschreibt die Studien sowie die damit verbundenen Orte und involvierte Organisationen.

- *Kategorie **Türkis**: Aufgaben und Termine*
  In diesem Schema werden Daten zur Administration von Aufgaben und Terminen beschrieben.

Da für den Sesam Export Manager in einer ersten Version lediglich Daten zu Experimenten, Prozessen und die daraus resultierenden wissenschaftlichen Daten sowie dazugehörige Datenbeschreibungen relevant sind, wird im Folgenden nur auf einzelne Entitäten der roten und grünen Kategorie eingegangen. Dafür werden sie einzeln aufgelistet, beschrieben und ihre Zusammenhänge untereinander in der nachfolgenden Abbildung 3 illustriert.

[4] Um Verwirrungen um den Gebrauch von statistischen Datenbanken auszuschliessen, wird an dieser Stelle festgehalten, dass für den Verwendungszweck der Daten unter sesam keine statistischen Datenbanken in Frage kommen. In sesam wird ein Mittel benötigt, um wissenschaftliche und administrative Daten zu speichern. Sie besitzen selbst keine Wahrscheinlichkeit, sondern sollen so wie sie sind als Fakten abgelegt werden.

- Die **ExperimentClass** beschreibt die Klasse, der ein **Experiment** (Messung eines Sachverhalts zu einem bestimmten Zeitpunkt für einen bestimmten Zweck an einem bestimmten Ort) angehört.
- Die **ProcessClass** stellt eine Prozessklasse dar. Ihr gehören einzelne Prozesse (**Process**) an.
- Die **ProcessOutputClass** bildet die Ausgabe eines Prozesses und somit ein Ergebnis eines gesamten Prozesses beziehungsweise eines einzelnen Arbeitsschritts. Es dient zur Unterscheidung möglicherweise gleicher Ausgabedaten aus einem Prozess und enthält ein Datenobjekt einer bestimmten DataItemClass.
- Der **ExperimentTypeWorkflowNode** verknüpft ein Experiment mit einer Prozessklasse, wobei gleichzeitig auch darin beteiligte Probanden und Informaten im **SubjectPriorityPath** definiert sind.
- Die **DataItemClass** beschreibt die Klasse eines Datenobjekts (DataItem).
- Die **DataItems** sind die eigentlichen Daten aus wissenschaftlichen Erhebungen wie Messungen, Fragebögen oder Beobachtungen.

In der untenstehenden Darstellung werden die oben aufgeführten Klassen (unten blau eingefärbt) in ihrem Zusammenhang aufgezeigt. Die Relationen werden als rosa Vierecke dargestellt.

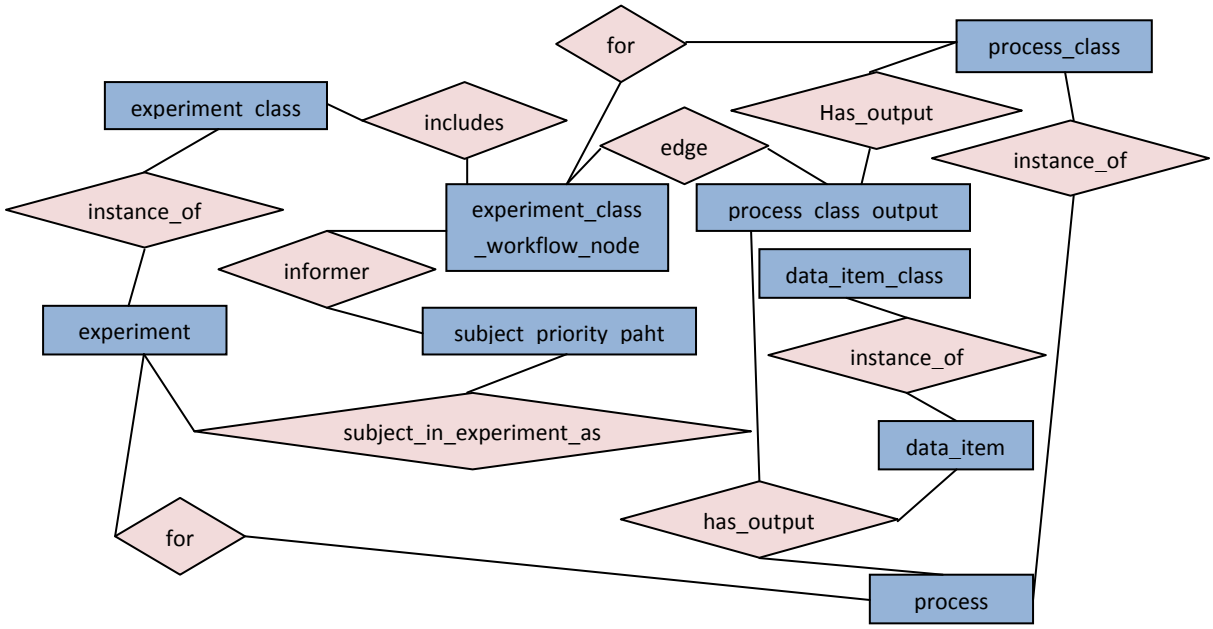

Abbildung 3: Ausschnitt aus dem Teilschema Prozesse und Experimente

In der sesamDB werden alle Datenänderungen mitgeführt. Dies ermöglicht es, den Stand der Datenbank für einen beliebigen Zeitpunkt wieder abzurufen. Für diesen Zweck wurden von den Entwicklern der sesamDB Gültigkeiten von Daten eingeführt wie auch eine Archivierungstabelle für jede Tabelle, die mittels Triggern und definierten Regeln gefüllt wird. Eine Änderung eines Datenobjekts beispielsweise löst automatisch eine Verschiebung des alten Datensatzes in die

Archivtabelle (mit Suffix _old) aus und speichert den neuen Datensatz an seiner Stelle in der aktuellen Tabelle (mit Suffix _now)[5].
Ausserdem kann in PostgreSQL Gebrauch von Triggern und eigens definierten Regeln gemacht werden.

[5] Um alle Daten (aktuelle und vergangene Werte) zu erhalten, gibt es in sesamDB die Sicht mit Suffix _all. Sie setzt sich zusammen aus _now UNION _old.

3 Verwandte Arbeiten

3.1 Gängige Statistiksoftware und assoziierte Datenformate

Mit aktueller Statistiksoftware ist es möglich, mit rechenintensiven Methoden grosse Datenmengen zu analysieren und statistisch auszuwerten. Die Auswahl an Statistikprogrammen ist riesig. Einige Programme wie SPSS und SAS sind inzwischen sehr weit verbreitet eingesetzt und somit zum Quasi-Standard geworden. Weitere häufig verwendete Statistikpakete sind STATA und R, auf diese wird jedoch an dieser aufgrund des Umfangs dieser Arbeit nicht weiter eingegangen.
Im Folgenden wird in Kapitel 3.1.1 und **Fehler! Verweisquelle konnte nicht gefunden werden.**zur Übersicht die Statistiksoftware SPSS und SAS erläutert und dabei auch die mit ihr assoziierten Datenformate vorgestellt. Zwar können alle erwähnten Programme Standardformate wie Excel-Arbeitsmappen, Access-Datenbanken oder CSV (Comma Separated Values) lesen, jedoch unterstützen diese Formate keine oder zu wenig Metainformationen über die Daten. Alle erwähnten Statistikprogramme werden als „Stand-Alone“ Anwendungen auf einem einzelnen Arbeitsplatz (PC oder Notebook) installiert und verwendet. Somit arbeitet der Benutzer jeweils selbständig mit seinen Daten.

3.1.1 SPSS

SPSS aus dem gleichnamigen Softwarehaus steht für Statistical Product and Service Solution und ist im Jahr 2007 in der Version 16 auf den Markt gekommen. Sie wurde in Java implementiert und ist deshalb auf Windows, Mac OS X wie auch Linux in der gleichen Version verfügbar. Bei SPSS (siehe Abbildung 4) handelt es sich um ein modular aufgebautes Programmpaket zur statistischen Datenanalyse und beinhaltet im Basismodul das grundlegende Datenmanagement sowie häufig verwendete statistische und grafische Datenanalysen. Über dies verfügt SPSS über eine Vielzahl weiterer Funktionen und Zusatzapplikationen wie OLAP, Data Mining und viele mehr.

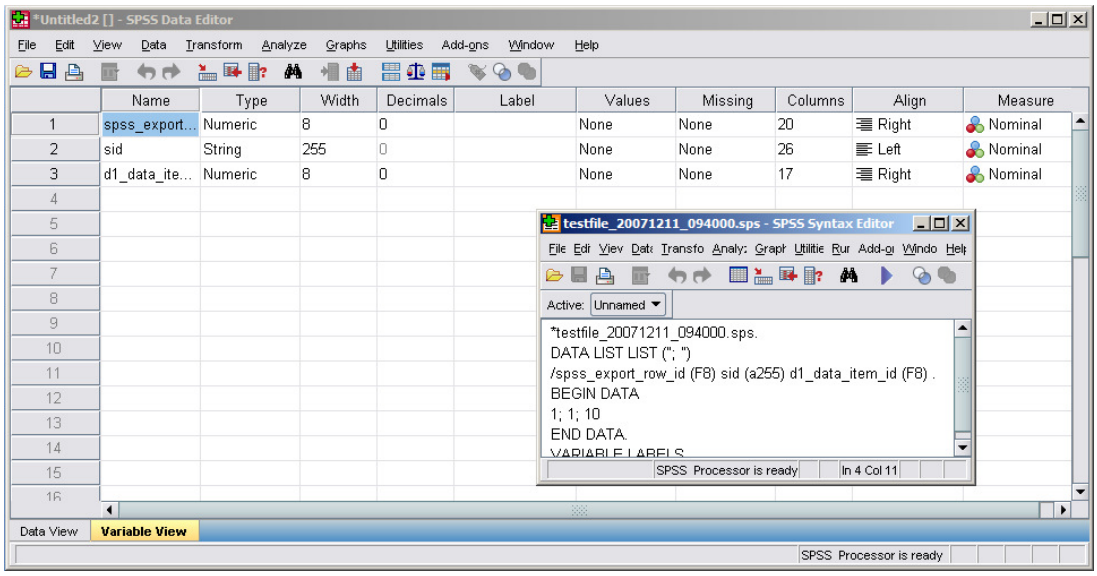

Abbildung 4: Screenshot von SPSS (Variable View und Syntax)

Das in SPSS standardmässig verwendete Datenformat ist .SAV und steht für SPSS Datensatz. Allerdings sind SAV-Dateien im Binärformat erstellt und somit nur mit dem System lesbar, welches die Datei auch erstellt hat, wie z.B. Windows. Müssen die Daten aber auch auf einem anderen System wie UNIX lesbar sein, so muss die Datei in das portable Format .POR konvertiert werden.
Das .POR Format ist zwar in ASCII geschrieben und kann somit problemlos von System zu System portiert werden. Ein Blick in die Datei mit Hilfe eines Editors macht allerdings rasch deutlich, dass dieses Format nicht intuitiv verstanden werden kann. Einfacher als .POR Dateien sind Syntax-Dateien (.SPS) zu verstehen (siehe auch kleines Fenster in Abbildung 4). Auch Variablen- und Wertebeschriftungen sowie weitere Variableneigenschaften können einfach hinzugefügt werden. Das Format unterstützt sowohl fixe Kolonnenbreiten wie auch Unterteilung durch Deliminiter. Zusätzlich können ebenfalls Gültigkeitsbereiche eingesetzt werden.
Das folgende Beispiel zeigt auf, wie in der Syntax-Datei erst die Datei selbst, danach das Aufzählungsformat und die Variablen definiert werden. Anschliessend folgen die eigentlichen Daten (Levesque, 2006, S. 38).

```
*testfile_20071211_094000.sps.
DATA LIST LIST (";")
/spss_export_row_id (F8) sid (a255) d1_data_item_id (F8).
BEGIN DATA
1; 1; 10
2; 1; 8
3; 4; 6
END DATA.
```

Abbildung 5: SPSS Syntax Datei mit Metadaten

Eine weitere Möglichkeit ist das Aufsplitten von Daten und Metainformationen über die Daten in je unterschiedliche Dateien (Levesque, 2006, S. 40).

```
*delimited_list.sps.
DATA LIST LIST
FILE='c:\examples\data\delimited_list.txt'
/id(F3) sex (A1) age opinion1 TO opinion5 (6F1).
EXECUTE.
```

Abbildung 6: SPSS Syntax Datei mit Metadaten und Verweis auf externe Datei

3.1.2 SAS

SAS vom amerikanischen Softwarehersteller SAS Institute ist ein Programmpaket, das ursprünglich für statistische Auswertungen entwickelt worden ist. SAS steht für Statistical Analysis Systems. Inzwischen wurde es jedoch stark ausgebaut und verfügt über viele andere Funktionen wie Datenerfassung, Datenhaltung, aber auch weitere Auswertungsmethoden wie Data Mining. Dank der langjährigen Kontinuität hat sich SAS vor allem im Pharma-Bereich zum Standardprodukt für die

Auswertung klinischer Studiendaten entwickelt. SAS ist sowohl für Windows wie auch UNIX und IBM z/OS verfügbar (SAS Institut, 2007).

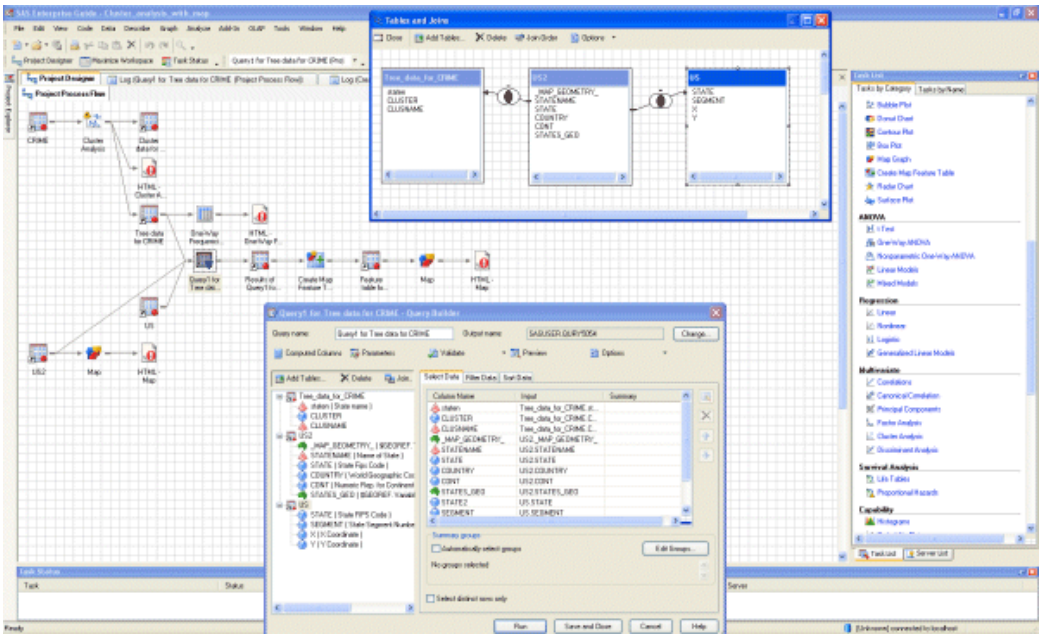

Abbildung 7: Screenshot von SAS (Clusteranalyse)

SAS verwendet als proprietäres Dateiformat SAS Tabellen (SAS data file). Ähnlich wie das SPSS-Format unterstützen SAS-Tabellen ebenfalls Metainformationen zu Daten. Auch SAS benutzt zum Einlesen von Daten eine Form von Syntax-Datei. Darin wird im Header erst die Datenmenge mit Namen und Variablenbezeichnungen beschrieben. Anschliessend folgen die eigentlichen Daten entweder in fixer Breite oder delimitiert durch ein beliebiges Zeichen (standardmässig mit einem Leerzeichen) (UCLA Academic Technology Services, 2007).

```
DATA cars;
  INPUT make $ 1-5 model $ 6-12 mpg 13-14 weight 15-18 price 19-22;
CARDS;
AMC  Concord2229304099
AMC  Pacer  1733504749
AMC  Spirit 2226403799
BuickCentury2032504816
BuickElectra1540807827
;
RUN;
```

Abbildung 8: SAS Data File mit Metadaten

Zusätzlich bietet sich die Möglichkeit an, Daten und Metainformationen in einzelne Dateien zu unterteilen. Im folgenden Beispiel werden nur die Metainformationen gespeichert. Die eigentlichen Daten befinden sich in der Datei cars.dat (UCLA Academic Technology Services, 2007).

```
DATA cars;
  INFILE 'D:\data\cars.dat';
  INPUT make $ 1-5 model $ 6-12 mpg 13-14 weight 15-18 price 19-22;
RUN;
```

Abbildung 9: SAS Data File mit Metadaten und Verweis auf externe Datei

3.2 Repräsentation von Datenbankanfragen für den Benutzer

3.2.1 Repräsentation in Skriptform

Die Standardanfragesprache im Bereich Datenbanken ist ohne Zweifel SQL[6]. Mit ihr können Datenbanken auf Inhalte durchsucht und diese dann ausgegeben werden. Die Syntax ist einfach, und semantisch ist SQL sehr nah an die englische Umgangssprache angelehnt. Eine Anfrage kann aus bis zu sechs Klauseln bestehen, wobei folgende Reihenfolge (siehe Abbildung 10) eingehalten werden muss und nur SELECT und FROM zwingend sind:

```
SELECT <Attribut- und Funktionsliste>
FROM <Tabellenliste>
[WHERE <Bedingung>]
[GROUP BY <Gruppierungsattribute>]
[HAVING <Gruppenbedingungen>]
[ORDER BY <Attributliste>]
```

Abbildung 10: Klauseln einer Anfrage in SQL

Im Folgenden werden nur auf die für diese Arbeit relevanten Anweisungen eingegangen. SQL selbst bietet jedoch weit umfangreichere Möglichkeiten.
Anfragen werden in SQL durch die Anweisung „SELECT“ gestartet (siehe Abbildung 11). Ihr folgt eine Liste mit Attributnamen, deren Werte mit der Ausführung der Anfrage aus der Datenbank gelesen werden. Um alle verfügbaren Attribute zu ermitteln, wird ein Stern (*) verwendet. Neben den vorhandenen Attributen können auch weitere abgeleitete oder eigendefinierte Attribute erzeugt werden (Elmasri & Navathe, 2005, S. 190). Hierzu werden Werte oder Operationen (z.B. Produkte oder Aggregationen wie SUM oder MAX) durch die Verwendung der Klausel „AS“ in einem neu benannten Attribut dargestellt (Elmasri & Navathe, 2005, S. 192).

[6] SQL ist eine Datenbanksprache zur Definition, Abfrage und Manipulation von Daten. Für den hier verfolgten Zweck ist jedoch nur die SELECT-Klausel von Bedeutung.

```
SELECT
       [Erweitertes Personal].Mitarbeitername,
       [Erweiterte Kunden].Kontaktperson,
       Verkaufschancen.Mitarbeiter AS [Zugewiesen an],
       Verkaufschancen.[Gesch Abschlussdatum],
       [Gesch Einnahmen]*[Wahrscheinlichkeit] AS Prognosewert
FROM
       (Verkaufschancen LEFT JOIN [Erweiterte Kunden] ON
       Verkaufschancen.[Kunde/Kundin] = [Erweiterte Kunden].ID) LEFT JOIN
       [Erweitertes Personal] ON Verkaufschancen.[Mitarbeiter] =
       [Erweitertes Personal].ID
WHERE
       ((Verkaufschancen.Geschlossen)<>True);
```

Abbildung 11: Beispiel einer Anfrage in T-SQL

Wie die obige Abbildung 11 veranschaulicht, dient die „FROM“-Klausel dazu, die Quelle der Daten zu definieren. Ihr folgt eine Liste mit Relationsnamen, auf die für die Ausführung der Anfrage zugegriffen wird. Dies können Tabellen- oder Anfragenamen sein, aber auch selbst wiederum vollständige Anfragen (Elmasri & Navathe, 2005, S. 198). Mittels Komma abgetrennte Quellen werden als kartesisches Produkt aufgerufen, die Anweisung „JOIN“ bildet einen Verbund.
In der „WHERE“-Klausel werden die Tupel eingeschränkt. Dabei handelt es sich um Bedingungen in Form eines boolschen Ausdrucks, unter denen die Daten ausgegeben werden sollen (Elmasri & Navathe, 2005, S. 193).
Weitere Möglichkeiten bieten Mengenoperatoren, mit denen die Anfragen um eine Anzahl Tubel erweitert oder gekürzt werden können. Hierzu gehören unter anderem UNION (vereint zwei Anfragen miteinander) und INTERSECT (bildet Schnittmenge).

Für den nicht erfahrenen Computerbenutzer ist diese Art von Datenbankanfrage viel zu schwierig, braucht es doch Erfahrung, um komplexe Anfragen über mehrere Tabellen durchführen zu können. Ausserdem fehlen dem Benutzer die Informationen über den Datenbankaufbau. Geeigneter wäre also eine grafische Repräsentation der vorhandenen Elemente, die er sich wie gewünscht aneinanderreihen könnte. Deshalb wird im folgenden Kapitel 3.2.2 näher auf diese eingegangen.

3.2.2 Grafische Repräsentation

Gewisse Datenbankmanagementsysteme bieten wie beispielsweise Microsoft Access ein grafisches Front-End an (siehe Abbildung 12). Sie ermöglichen dem Benutzer, sich ohne Kenntnisse über den Aufbau der Datenbank oder über eine Datenbanksprache, Anfragen zu erstellen.

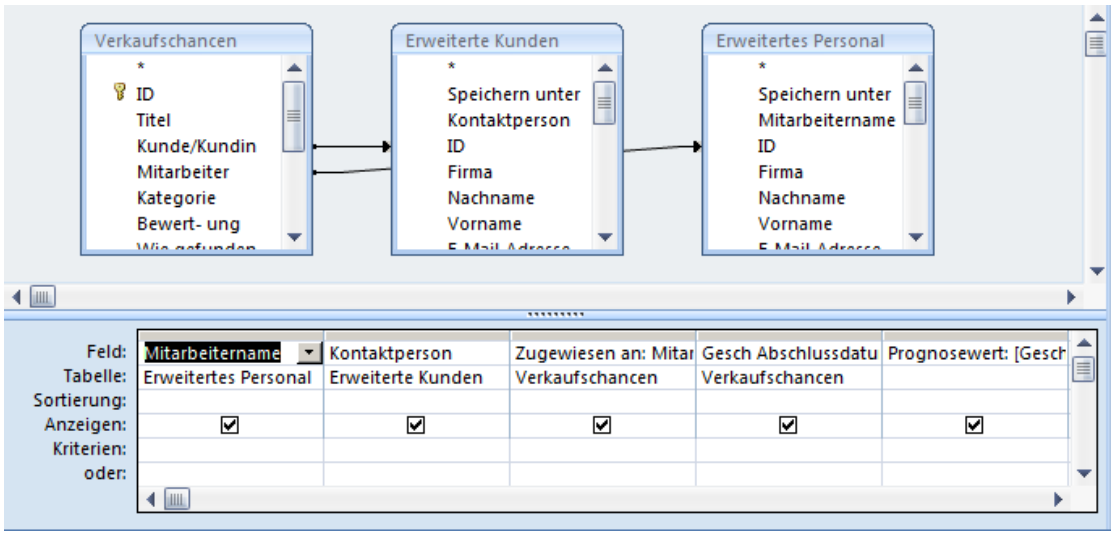

Abbildung 12: Screenshot aus Microsoft Access (grafische Repräsentation einer Anfrage)

Dies gilt natürlich nur solange, wie nicht zu viele Auswahlmöglichkeiten zur Verfügung stehen und diese eindeutig und übersichtlich benannte sind. In grossen Datenbanken mit hunderten von Tabellen wird jedoch dies rasch unübersichtlich. Der Benutzer kann sich nicht mehr zurechtfinden. Häufig benötigt der Benutzer auch nicht die gesamte Datenmenge einer Datenbank, wie einzelne Attribute, z.B. die letzte Änderung einer Tabelle.
Die Daten müssen also für den ungeübten Benutzer ohne das Dazulernen von Konzepten oder Sprachen abrufbar sein. Da aber keines dieser beiden eben vorgestellten Konzepte den Anforderungen eines Benutzers der sesamDB genügen würde, ist hier eine eigene, individuelle Umsetzung nötig.

3.3 Anonymität und Pseudonymität

Für die Auswertung medizinischer Daten für Forschungszwecke ist ein direkter Personenbezug nicht nötig. Damit verschiedene Daten eines Falles zusammengeführt oder neu erhoben werden können, werden Personenangaben jedoch trotzdem oft mitgeführt (Pommerening, 2000, S. 1). Gemäss Art. 19a des Datenschutzgesetzes müssen Personendaten jedoch für statistische Auswertungen anonymisiert werden, sobald dies der Verwendungszweck erlaubt (Bundesstatistikgesetz, 2007).
Unter Anonymität versteht man gemäss dem Deutsche Wörterbuch Wahrig die„Verschweigung, Nichtangabe des Namens“ (Wahrig, 1997). Somit ist Anonymität im Falle von statistischen Auswertungen die Geheimhaltung der Identität einer Person. Dies umzusetzen ist im Alltag beinahe unmöglich, können doch einzelne Angaben über eine Person ihre Identität aufdecken (wie z.B. der Schweizerischer Bundesrat, abgewählt am 12. Dezember 2007). Somit ist gemäss Rost der Grad der Anonymität zentral, also inwiefern es möglich ist, aufgrund von Einzelattributen auf eine Identität zu schliessen. Je nach Grad der Anonymität entscheidet man in der Statistik folgende drei Stufen (2003):

- Formale Anonymität:
  Alle Namen sind zwar entfernt, die anderen Daten bleiben jedoch unverändert bestehen. Die Daten können relativ leicht einer Identität zugeordnet werden.

- Faktische Anonymität:
  Die Daten sind nur mit unverhältnismässig grossem Aufwand zuordenbar.

- Komplett Anonym:
  Die Zuordnung zu einer Identität ist nicht möglich.

Zu veröffentlichende Statistikdaten sind wie oben bereits festgehalten so weit wie möglich zu anonymisieren, d.h. die personenbezogenen Daten müssen soweit verändert werden, dass sie nicht mehr einer Person zugeordnet werden können. Eine einfache Möglichkeit dazu ist die Pseudonymisierung.
Bei der Pseudonymisierung wird gemäss Pfitzmann das Identifikationsmerkmal (in der Regel ein Name) durch ein Pseudonym[7] ersetzt (2004, S. 9). Auf diese Wiese soll der Nachweis der Personenbezug so verschleiert werden, dass faktische Anonymität entsteht. Die Pseudonymisierung hat den Vorteil, dass Bezüge verschiedener Datensätze, die auf die gleiche Weise pseudonymisiert wurden, erhalten bleiben. Je nach Art der Erzeugung der Pseudonyme können hier nach Pommerening unterschiedliche Typen definiert werden (2000, S. 1):

- Deterministische Pseudonyme:
  Die Erzeugung des Pseudonyms erfolgt über eine schlüsselabhängige Hashfunktion aus Identitätsdaten durch eine vertrauenswürdige Instanz.

- Willkürliche Pseudonyme:
  Der Benutzer erzeugt ein Pseudonym in einem Einmal-Algorithmus aus einem Geheimnis, wie beispielsweise einer Passphrase.

- Zufällige Pseudonyme
  Das Pseudonym wird zufällig mittels eines Zufallsverfahrens erzeugt oder wird frei gewählt. Auf diese Weise erzeugte Pseudonyme eignen sich für den einmaligen Gebrauch, wie zur Zusammenführung verschiedener Datenquellen zu Statistikzwecken. Sonst macht sie nur Sinn, wenn sie in einer Referenzliste gespeichert wird.

Je nach Anwendungsgebiet ist die geeignete Form von Pseudonymen zu wählen.

Die Einführung von Pseudonymen stellt einen Kompromiss zwischen der Weiterverwendbarkeit von Daten und datenschutzrechtlichen Überlegungen dar. Kryptische Pseudonyme stellen somit „eine Grundtechnik des praktischen Datenschutzes dar“, so Pommerening (2000) und sollten deshalb „wo immer möglich eingesetzt werden“.

[7] Ein Pseudonym ist gemäss dem Deutschen Wörterbuch Wahrig ein Deckname für eine Entität (Wahrig, 1997).

3.4 Aufbau von modellbasierten Anwendungen

Das Architekturmuster Model-View-Controller (MVC) dient dazu, Softwaresysteme in drei Einheiten, nämlich das Model (Datenmodell), die View (Präsentation) sowie den Controller (Programmsteuerung) aufzuteilen (siehe Abbildung 13) (Middendorf, Singer, & Heid, 2002).

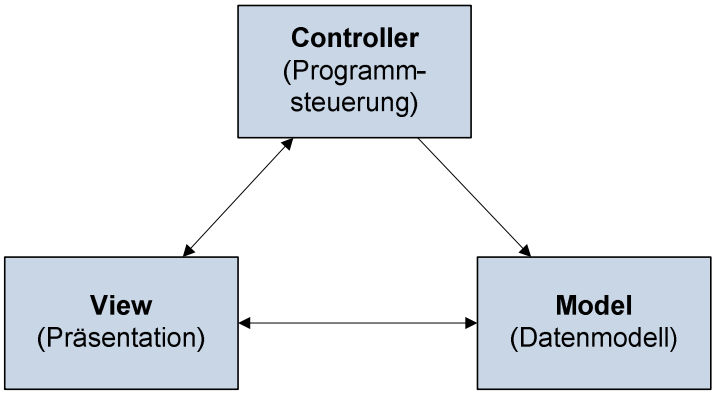

Abbildung 13: Design Pattern Model View Controller (MVC)

Das *Datenmodell* repräsentiert laut Middendorf et al. die darzustellenden Daten sowie die Geschäftslogik. Die Daten werden unabhängig vom Erscheinungsbild bereitgestellt. Die *Präsentation* dient zur Programmsteuerung durch den Benutzer und stellt Inhalte für ihn dar. Durch die Unabhängigkeit der Präsentation ist es meist relativ einfach möglich, weitere Benutzeroberflächen wie beispielsweise eine GUI sowie eine Konsolenbedienung zu erstellen. Die Einheit *Controller* ist die eigentliche Programmsteuerung. Sie nimmt Benutzeranweisungen entgegen und agiert entsprechend. Der Vorteil einer Aufteilung in diese drei Einheiten liegt vor allem in der Wiederverwendbarkeit einzelner Programmkomponenten und vereinfacht Programmerweiterungen und Änderungen (2002).

4 Analyse der Anforderungen an den Sesam Export Manager

In diesem Kapitel werden die Anforderungen an die zu entwickelnde Exportanwendung, den Sesam Export Manager, detailliert dokumentiert. Die Aufgabenstellung der Bachelorarbeit gibt zwar hier einen groben Rahmen vor, konkretisiert wurden die Spezifikationen jedoch in mehreren ausführlichen Gesprächen mit meinem Betreuer Boris Galvic[8]. Das Hauptziel der Anwendung ist es, Daten der sesam Datenbank so zu extrahieren, dass diese in ein gängiges Statistikprogramm eingelesen werden können.
Der gesamte Zielprozess kann dabei in drei Teile unterteilt werden, wie in der nachfolgenden Abbildung ersichtlich ist:

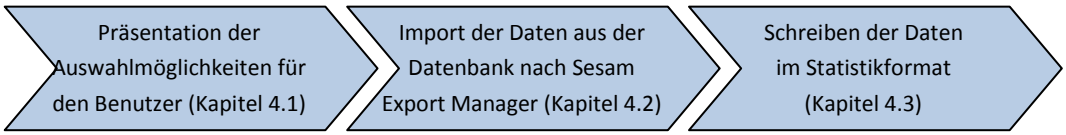

Abbildung 14: Export von Daten aus sesamDB zur Verwendung in einem Statistikprogramm

4.1 Präsentation der Auswahlmöglichkeiten

Um dem Benutzer die Möglichkeit zu geben, Daten aus der sesamDB auszulesen, muss dieser erst wissen, welche Daten verfügbar sind. Dem Anwender, meist Psychologen und Soziologen, jedoch auch Gynäkologen, Geburtshelfer, Psychiater, Genetiker, Pharmakologen, Biologen und Vertreter anderer Disziplinen, kann allerdings nicht zugetraut werden, sich mit SQL auszukennen, geschweige denn das Datenbankschema zu verstehen. Der komplizierte Aufbau der Datenbank mit Vererbung erschwert das Zurechtfinden in den Datenbankinhalten zusätzlich. Des Weiteren sollen unter anderem, auch aufgrund Datenschutzbestimmungen, gewisse Aspekte gegen aussen verborgen bleiben. Damit der Benutzer trotzdem für seine Bedürfnisse sinnvolle Anfragen erstellen kann, muss er dabei vom Programm geführt werden. Ein Standard-Datenbankanfrageprogramm kommt deshalb nicht in Frage, es bedarf einer für diesen Zweck spezialisierten Anwendung. Ein Beispiel soll dies verdeutlichen: Der Benutzer möchte die Ergebnisse eines CIDI[9]-Interviews mit Müttern, die zum Zeitpunkt des Interviews in der 20. Schwangerschaftswoche waren, erhalten.
Dem Benutzer muss also präsentiert werden, welche Datenobjektklassen zur Verfügung stehen (CIDI-Fragebogen), um welches Experiment es sich dabei handelt (CIDI Fragebogen ausfüllen), welcher ProzessOutput dabei genau benötigt wird (Resultate des ausgefüllten Tests) und schlussendlich wer den Fragebogen ausgefüllt hat (die Mutter). Über dies wird eine Bedingung benötigt, die alle Ergebnisse der Anfrage auf Mütter der 20. Schwangerschaftswoche einschränkt.

[8] Boris Glavic ist Assistent in der Database Technology Research Group am IFI (Institut für Informatik der Universität Zürich) und ist bei sesamDB Project Leader.
[9] Beim CIDI, Clinical Interview Diagnosis Instrument, handelt es sich um ein standardisiertes klinisches Interview mit genau festgelegten Ablauf- und Auswertungsregeln. Je nach Beantwortung der Fragen ändert im CIDI der Ablauf des Interviews. Für Auswertungen werden oft Aggregationen gemacht, wie z.B. ein Depressions-Score.

Ein etwas komplizierteres Beispiel ist die Frage nach den Resultaten eines Bluttests der Mutter in einem Stress-Experiment mit mehreren Bluttests, wie beispielsweise einem Bluttest vor und einem nach dem Stresstest (siehe Abbildung 15). In diesem Beispiel wird der Wert vor dem Stresstest gesucht.

Abbildung 15: Bluttest vor und nach Stresstest

In diesem Fall muss die Art der Probe (also Bluttest der Mutter vor einem Stresstest), das Experiment (Stress-Test), der Prozess-Output (Blutwert) sowie der Proband (Mutter) angegeben werden. Zusätzlich muss der Workflow Node (Bluttest vor dem Stresstest) definiert werden, d.h. an welcher Stelle des Experiments der Test erfolgt.
Anfragen können beliebig kompliziert werden, indem beispielsweise mehrere verschiedene Probanden (Mütter und Väter) ausgewählt, weitere Bedingungen (z.B. nur Probanden aus Basel) hinzugefügt oder gar eine Vielzahl von Datenobjekten (Blutwert von Adrenalin und Körpertemperatur) verknüpft (z.B. als JOIN) ausgewertet werden müssen.
Um den Benutzer der Exportapplikation nicht zu überfordern, muss deshalb abgeschätzt werden, nach welchen Datenobjekten er am ehesten suchen wird. Die meist benutzte Funktion wird gemäss Gespräch mit Boris Glavic vom 7. Dezember 2007 die Suche nach Ergebnissen aus Experimenten sein. Aus diesem Grund wird ihr die höchste Priorität zugemessen. Weitere mögliche Suchstrategien sind die Suche nach Probanden (z.B. zum Vergleich einzelner Testwerte) oder nach Instrument (z.B. zum Vergleich der Testgenauigkeit).
Die Auswahl der für die Suche benötigten Einschränkungsparameter sollte, um den Benutzer nicht mit Anfragesprachen oder Datenbankaufbau zu belasten, über einer für seine Zwecke zugeschnittenen, gut strukturierten grafischen Benutzeroberfläche erfolgen.

Um eine Eindeutigkeit bei der Definition der Datenanfragen zu erzielen, müssen unten stehende Auswahlkriterien in der grafischen Benutzeroberfläche vorhanden sein (eine Erklärung der unten aufgezählten Klassen ist im Kapitel 2.5 zu finden):

- DataItemClass: Klasse, der die Datenobjekte angehören.
- ExperimentClass: Experimentklassen
- ProcessClassOutput: Ausgabe von Prozessen
- ExperimentTypeWorkflowNode: Verknüpft Experimente mit Prozessen
- SubjectPriorityPath: Ordnet dem Ablauf die Rollen der beteiligten Personen zu.

Wäre die Anfragedefinition nicht eindeutig, so würden zu einem mehrmals durchgeführten Test je alle Erhebungsdaten zurückgegeben. Um die Ergebnisse eines genau definierten Tests zu erhalten, müssen deshalb alle Anfrageparameter gesetzt sein.

Ein weiteres Anwendungsbeispiel ist das Abrufen von Daten aus der sesamDB, wie sie zu einem früheren Zeitpunkt vorhanden waren. Dies findet häufig dann Verwendung, wenn beispielsweise

Journalartikel veröffentlicht und erst danach Daten korrigiert werden. Wird nun von Reviewern eine Ausweitung der Studie auf weitere Attribute erwartet, so ist dies nicht mehr möglich, da die Daten geändert haben können. Ein Beispiel dafür ist die Änderung von Daten durch Qualitätssicherung.
Dank der Verwendung eines temporalen Datenbankaufbaus sind alle Daten inklusive deren Änderung für jeden Zeitpunkt verfügbar. Diese Einschränkung des Exports auf einen in der Vergangenheit liegenden Zeitpunkt sollte deshalb in Sesam Export Manager umgesetzt werden.
Hat der Anwender schon einige Anfragen erstellt, so ist er vielleicht daran interessiert, diese zu speichern und bestehende Anfragen wiederzuverwenden oder zu modifizieren. Fertige Anfragen sollten deshalb in der Datenbank gespeichert oder als Datei abgelegt, und danach wieder ins Programm geladen werden können. Die Anwendung sollte demnach einen Anfrageverlauf anbieten, welche den Zeitpunkt und den Aufbau von Anfragen beinhaltet oder speichert, um weitere Anfragen erneut genauso oder verändert auszuführen.

4.2 Import der Daten aus der Datenbank nach Sesam Export Manager

Die durch den Benutzer über das GUI festgelegten Anfrageparameter bilden im nächsten Schritt die Grundlage für die Generierung der entsprechenden konkreten SQL-Anfrage. Diese soll genau die Daten zurückgeben, die der Benutzer aufgrund seiner Auswahl erwartet.

4.3 Schreiben der Daten im Statistikformat

Nachdem die Daten im Sesam Export Manager vorhanden sind, müssen diese in eine Datei im benötigten Format geschrieben werden. Als geeignetes Format für Statistikdaten hat sich in einem ersten Schritt das SPSS Syntax File herauskristallisiert, da es erstens durch eines der meist verwendeten Statistikprogramme lesbar ist, da es zweitens als nicht-binäres Format relativ leicht zu generieren ist und drittens die Daten darin ausführlich beschrieben werden können. In Kapitel 3.1.1 wurde dieses Format bereits genauer diskutiert. Der Sesam Export Manager sollte jedoch in der Lage sein, relativ einfach um weitere Exportformate erweitert zu werden. Diese sollte der Benutzer auf der grafischen Oberfläche auswählen können.
Der Umfang der zu exportierenden Daten soll sich allerdings nicht nur auf die effektiven Messdaten beschränken, sondern sollte auch die dazugehörigen Metadaten beinhalten. Zu diesen gehören:

- Variablennamen (Name der zu exportierenden Spalte)
- Datentyp (z.B. Integer, Text, Datum, etc.)
- Variablenbeschriftung (Beschreibung der Variablen, damit sich der Benutzer unter einem Variablennamen, meist einem Kürzel, etwas vorstellen kann, z.B. Skala von 1-5)

Weitere sinnvolle Parameter sind:

- Skalentypen (verwendet für statistische Berechnungen)
- Wertebeschriftungen (werden verwendet, um etwas über den Wert auszusagen. Wert 5 entspricht beispielsweise gut).

Neben dem Umfang des Exports ist es auch nötig für hinreichende Anonymität für die an der Studie beteiligten Personen zu sorgen. Um ein ausreichendes Mass an Datenschutz gewährleisten zu können, müssen alle personenspezifischen Daten vor dem Export in die Statistikdatei pseudonymisiert werden (vgl. hier zu Kapitel 3.3).

4.4 Zusammenfassung der Anforderungen

Zusammenfassend werden nachfolgend nochmals alle Anforderungen aus den vorhergehenden drei Unterkapiteln an Sesam Export Manager aufgelistet und nach Prioritäten sortiert:

Unverzichtbare Funktionalitäten

- Grafische Benutzeroberfläche mit Repräsentation der verfügbaren Daten
- Generierung einer Datenbankanfrage über GUI
- Export von Daten aus sesamDB in durch Statistikprogramm lesbares Format
- Pseudonymisierung Personenspezifischer Daten
- Einfache Erweiterbarkeit um neue Exportformate
- Implementierung in Java
- Installationsanweisung
- Dokumentation der Anwendung
- Testen der univerzichtbaren Funktionalitäten

Wichtige Funktionalitäten

- Export von Metadaten zu den ausgewählten Daten
- GUI für Eingabe des Benutzerlogins
- Auswahl der Datenbank über GUI
- Auswahl des Exportplugins über GUI
- Leichte Anpassung der Applikation bei Änderung der Datenstruktur von sesamDB
- Testen der wichtigen Funktionalitäten

Hilfreiche Funktionalitäten

- Verlaufsübersicht von Datenbankanfragen
- Speichern und Laden von Datenbankanfragen inklusive Dokumentation
- Zugriff auf Datenbestände eines früheren Zeitpunkts
- Testen der hilfreichen Funktionalitäten

5 Entwurf und Umsetzung des Sesam Export Managers

Dieses Kapitel beschreibt in den folgenden vier Teilkapiteln den Aufbau des Sesam Export Managers. Im Kapitel 5.1 wird aufgezeigt, aus welchen Komponenten die Applikation besteht, für was die einzelnen Komponenten zuständig sind, wie sie zusammenspielen und warum sie so umgesetzt werden. Das Kapitel 0 enthält einen Detailentwurf zu einzelnen Kernkomponenten. Diese werden genau analysiert, womit der bewusst abstrakt gehaltene Teil über den konzeptionellen Aufbau aus Kapitel 5.1 verfeinert wird. In Kapitel 5.3 werden die zur Umsetzung benötigten Hilfsmittel wie beispielsweise die Datenbankschnittstellenkomponente diskutiert sowie deren Vor- und Nachteile abgewogen.

5.1 konzeptioneller Aufbau

In diesem Kapitel folgt eine Beschreibung des konzeptionellen Aufbaus der Anwendung. Dazu wird in einem ersten Teil ein Gesamtüberblick über die verwendete Architektur gegeben, im zweiten Teil werden die einzelnen Schichten genauer erläutert sowie die darin enthaltenen Komponenten diskutiert.

5.1.1 Gesamtüberblick über den konzeptionellen Aufbau

Sesam Export Manager ist als Einzelplatzanwendung aufgebaut, das heisst, sie kann auf einem Einzelplatzrechner installiert und ausgeführt werden. Dies ist sinnvoll, weil sich die Datenbank geschützt und gegen aussen abgeschlossen in der Sesam Zentrale befindet. Ein Zugriff über Internet ist also nicht möglich, sondern die Daten können nur vor Ort an einer an die sesamDB angeschlossene Arbeitsstation abgerufen werden. Wäre die Datenbank durch das Internet erreichbar, so würde sich eine Webanwendung eignen. Allerdings würde dies zu einem erhöhten Wartungsaufwand führen, da zusätzlich ein Webserver nötig wäre und bedeutend höhere Sicherheitsvorkehrungen getroffen werden müssten. Nachteilig wirkt sich dagegen die Versionsverwaltung für die Einzelplatzanwendung aus, da jede installierte Version einzeln bei einem neuen Release auf den neusten Stand gebracht werden muss. Da jedoch eine sesamDB-Updateanwendung existiert, mit der beim Aufstarten des Rechners auf neue Versionen geprüft wird, ist der Mehraufwand dafür relativ gering. Die Aktualisierungssoftware kann in die Exportapplikation eingebaut werden und übernimmt die Arbeit für das allfällige Update.
Eine weitere Möglichkeit wäre die Umsetzung als Erweiterung für eine bereits vorhandene Statistikanwendung. Dies würde es zwar für den Benutzer der einzelnen Statistikanwendungen komfortabel gestalten, Daten zu importieren, hätte jedoch auch einige Nachteile. Zum einen müsste für jede Statistikanwendung eine eigene Erweiterung geschrieben werden, was einen enormen Aufwand bedeuten würde. Aufgrund mangelnder Schnittstellen wäre dies nicht überall möglich und die Exporte würden sich auf wenige Statistikformate beschränken, was wiederum den Anforderungen an die zu wählende Lösung wiederspricht.

Im Gegensatz zur Verwendung eines monolithischen Aufbaus, in der alle Komponenten ineinander verzahnt sind, ist gemäss dem Model-View-Controller Ansatz aus Kapitel 3.4 der sesam Export Manager in drei Teile aufgeteilt. Er enthält die grafische Benutzeroberfläche, die Main Application (Steuerung) sowie das darunterliegende Datenmodell. Die nachfolgende Abbildung 16 stellt die Architektur der Applikation schematisch gegliedert dar.

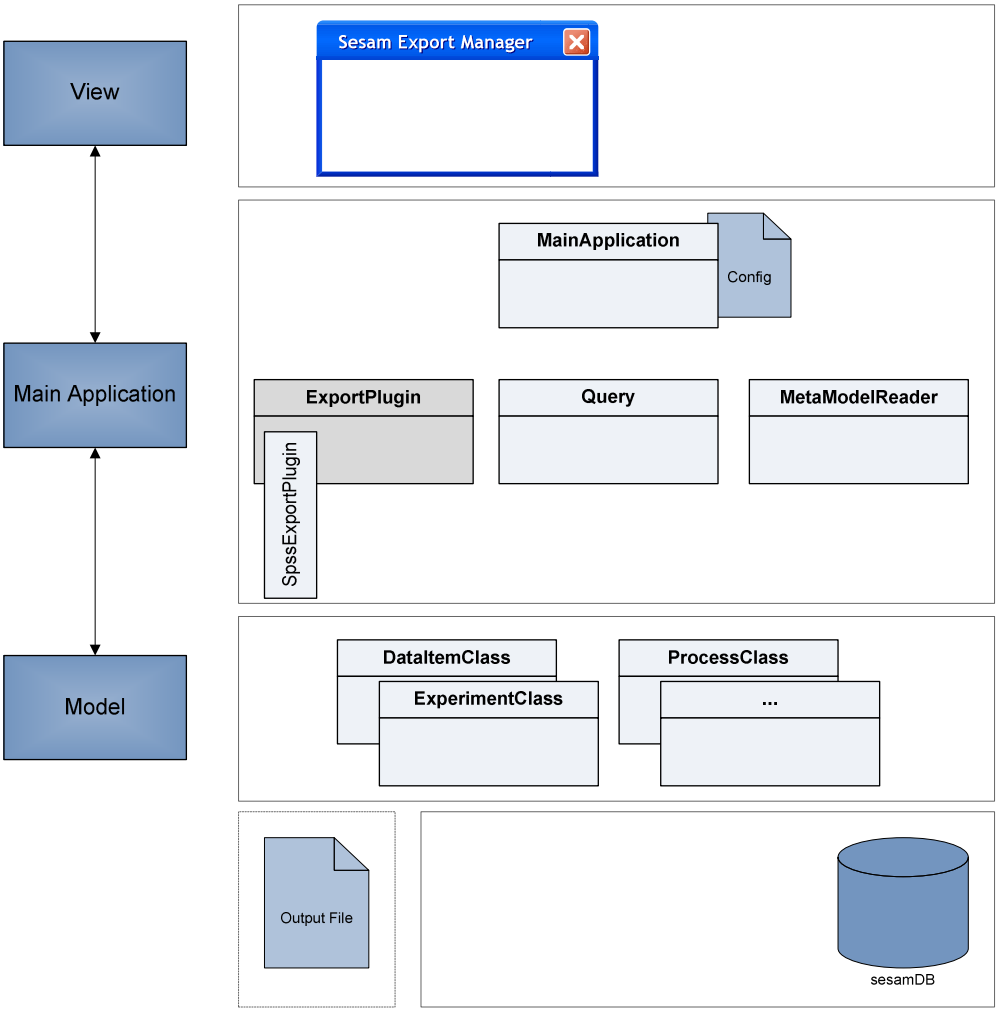

Abbildung 16: Aufbau des Sesam Export Managers

Dieser in Komponenten aufgeteilte Aufbau bietet eine Reihe von Vorteilen gegenüber einer verzahnten Implementierung. Zum einen wird der Code übersichtlicher und einfacher nachvollziehbar, da jeder Teil für sich einzeln betrachtet werden kann. Die einzelnen Komponenten sind nicht ineinander gekoppelt und beeinflussen sich somit gegenseitig nicht. Dies bietet ausserdem den Vorteil, dass spätere Änderungen und Erweiterungen leichter umgesetzt werden können, da nicht das gesamte Programm verstanden werden muss, sondern lediglich die zu ändernden Stellen. Eine Kapselung erhöht ausserdem die Wiederverwendbarkeit einzelner Programmteile. Besonders bewährt hat sich dieser Aufbau für Erweiterungen. Beispielsweise wäre eine Erweiterung der Applikation um eine weitere Benutzerschnittstelle wie eine Konsole relativ einfach umsetzbar, da keine Funktionalität zur Programmsteuerung in der Benutzeroberfläche, sondern nur in der

Steuerung verankert ist. Ausserdem ist das Testen in einer MVC-Architektur bedeutend einfacher, da die einzelnen Teile so für sich abgeschlossen auf ihre richtige Funktionstätigkeit geprüft werden können und sich nicht gegenseitig beeinflussen sollten.
Natürlich könnte die Applikation auch ohne Steuerungsklasse funktionsfähig aufgebaut werden. Dies gäbe jedoch keinen nennenswerten Vorteil, da die Methoden nur in andere Klassen verlagert und sich der Implementierungsaufwand damit nicht verringern würde. Allerdings hätte dies starke Auswirkungen auf die Erweiterbarkeit und die Testbarkeit, wie oben begründet.

5.1.2 Konzeptioneller Aufbau der einzelnen Schichten

Wie im letzten Kapitel behandelt, werden an dieser Stelle die folgenden drei Schichten gemäss des MVC-Ansatzes behandelt:

- Benutzeroberfläche
- Modelschicht
- Steuerungsschicht

Die erste Schicht, die *Benutzeroberfläche*, dient zur Entgegennahme von Benutzerinteraktionen mit dem Sesam Export Manager. Dies sind beispielsweise die Auswahl der zu exportierenden Daten oder der Anstoss zum Ausführen einer Anfrage. Die Benutzeroberfläche selber ist wiederum in einzelne Elemente aufgeteilt. Dies erhöht die Übersichtlichkeit, erleichtert das Testen und macht es möglich, dass einzelne Teile wiederverwendet werden können. Benutzerbefehle werden durch eigene Aktionsklassen repräsentiert. So kann die Benutzeroberfläche vollständig von der Steuerung getrennt werden.

Die *Modelschicht* repräsentiert grösstenteils Daten aus der Datenbank. In ihr sind alle für die Applikation relevanten Datenbankinhalte wie beispielsweise die ExperimentClass (vgl. Kapitel 2.5) oder ProcessOutputClass auf ein eigenes Objekt abgebildet und bereits benötigte Daten aus der Datenbank in den Hauptspeicher des Clientrechners geladen. Jedoch ist es nicht sinnvoll, gleich die gesamte Datenbank mit allen Inhalten zu laden, da dies erstens bei der grossen Datenmenge sehr lange dauert und viel Speicher benötigt, aber auch weil ein Grossteil der Daten gar nie für den Export der Daten verwendet würde. Mittels der Klasse MetaModelreader aus der Steuerungsschicht werden die gleich benötigten Daten direkt beim Programmstart beziehungsweise der Verbindung mit der Datenbank geladen. Diese Klasse benutzt einen Hauptspeicherpuffer, der die geladenen Daten zwischenspeichert. Auf diese Daten können dann direkt zugegriffen werden, noch nicht im Hauptspeicher vorhandene Daten werden bei Bedarf nachgeladen.
Ebenfalls denkbar wäre es, die anzuzeigenden Daten jeweils bei Bedarf aus der Datenbank direkt zu holen. Dies würde aber zu einer hohen Anzahl Zugriffe auf die Datenbank führen und somit erstens die Datenbank belastet und zweitens aufgrund der Verzögerung durch den entfernten Zugriff den Benutzerkomfort einschränken.

Die *Steuerungsschicht* verwaltet alle Abläufe im Sesam Export Manager und wird beim Start der Applikation als erstes aufgerufen. Sie lädt zu Beginn alle Programmeinstellungen, wozu in einer

ersten Umsetzungsphase auch die Datenbankeinstellungen wie URL, Port, Datenbankname, Benutzername und Passwort gehören. Später sollen diese Angaben jedoch durch eine Eingabe beim Start des Programms zur Verfügung stehen. Mit diesen Informationen stösst die Applikation dann die Abbildung der relevanten Datenbankinhalte auf die Klassen im Modell an. Abschliessend wird die Benutzeroberfläche gestartet und die aus der Datenbank geladenen Daten darin dargestellt.
Nach einer Benutzerinteraktion nimmt die Steuerungsklasse den Befehl entgegen und agiert entsprechend. Wurden alle zur Definition benötigten Anfrageparameter eingegeben, so wird durch das Auslösen einer Aktion eine Query gestartet. Diese Query erhält alle Anfragewerte, den Dateinamen sowie das gewünschte Exportformat, lädt die Daten aus der Datenbank und speichert die abgefragten Datenbankinhalte mithilfe des Export-Plugins im entsprechenden Dateiformat. Der Aufbau des Exportprozesses wurde mithilfe von Plugins konzeptioniert, da auf diese Weise einfach um weitere Exportformate und Exportmethoden erweitert werden kann. Ohne diesen Ansatz wären die Anfrage und der Export der Datenbankinhalte nicht sauber von einander getrennt, was zu Problemen bei der Definition weiterer Exportformate führen würde. Es müssten dann nicht nur Export-, sondern auch Anfragefunktionalität neu programmiert werden.

5.2 Detailentwurf

Dieses Kapitel beschreibt den Detailentwurf der Applikation. In den folgenden Unterkapiteln wird der Reihe nach erst auf die Konfiguration des Sesam Export Managers in Kapitel 0 eingegangen und erläutert, wie diese festgelegt und geladen wird. Anschliessend folgt in Kapitel 5.2.2 ein ausführlicher Entwurf der Benutzeroberfläche. Das Kapitel 5.2.3 behandelt, wie die Metamodelldaten aus der Datenbank geladen werden. Ferner wird in Kapitel 5.2.4 auf den Export der Studien- und Metadaten in Statistikdateiformate eingegangen, wobei in Kapitel 5.2.5 genau aufgezeigt wird, wie die SQL Anfrage generiert wird. Schlussendlich folgt in Kapitel 5.2.6 der Pseudonymisierungsmechanismus.

5.2.1 Konfiguration des Sesam Export Managers

Die Konfiguration des Sesam Export Managers erfolgt über eine XML-Datei in einem für Ressourcen reservierten Ordner innerhalb der Applikation. Darin werden Einstellungen wie der Datenbankname oder der Datenbanktyp gespeichert und mit einer Klasse zum Einlesen von XML-Dokumenten geladen. Damit Einstellungen auch nach der Auslieferung des Programms angepasst werden können, wird die Konfigurationsdatei nicht in ein JAR eingebunden, sondern als einzelne Datei in einem Ressourcenordner abgelegt.

5.2.2 Benutzeroberfläche

Die GUI des Sesam Export Managers benutzt das SWT-Framework wie dies gemäss den Anforderungen in Kapitel 4.1 vorgegeben ist und macht zusätzlich Gebraucht von JFace (siehe Kapitel 5.3). Das Hauptfenster ist in zwei Teile eingeteilt. Es enthält einen Einstellungsbereich (siehe grüner Kasten in Abbildung 17) sowie einen Anfragebereich (roter Kasten in Abbildung 17). Implementiert werden diese zwei Bereiche als eigene Klassen, wodurch die GUI übersichtlich implementiert wird und einzelne Elemente wiederverwendet werden können.

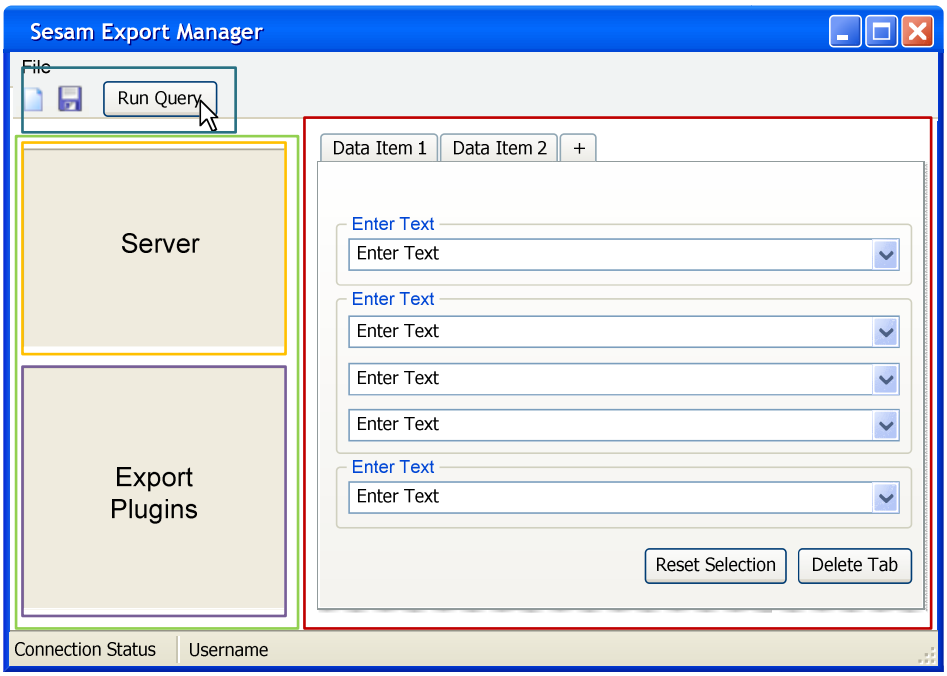

Abbildung 17: Entwurf der Benutzeroberfläche

Im Einstellungsbereich auf der linken Seite des GUIs (grüner Kasten) wird für jede Einstellungskategorie eine eigene neue Komponente darin erstellt. Beispiele dafür sind bekannte Server (gelber Kasten) oder auswählbare Export-Plugins (violetter Kasten). Auf Dauer kann diese Leiste um weitere Einstellungsmöglichkeiten erweitert werden wie beispielsweise eine Auswahl bereits erstellter Exporte oder eine Datumseinstellung zur Anfrage der Datenbank zu einem früheren Zustand.

Im Anfragebereich werden Anfragen durch das Hinzufügen einzelner Ergebnisse aus Experimenten, repräsentiert durch Tabs, definiert. Diese Elemente werden beim Start der Applikation geladen. Neue Experimente, etc. in der Datenbank werden erst nach dem nächsten Neustart der Applikation geladen. Dass Daten benötigt werden, die gerade zum Zeitpunkt der Ausführung des Sesam Export Managers geladen werden, ist aber eher unwahrscheinlich und somit rechtfertigt dies den geringeren Implementierungs-, wie auch Ausführungsaufwand. Ansonsten müssten ständig Datenbankanfragen starten, um die Datenbank auf Änderungen zu überprüfen.

Die Selektion erfolgt top-down, indem von oben bis unten nach einer Auswahl die verfügbaren Elemente der darunterliegenden Elemente immer weiter eingeschränkt werden. Die dazu nötigen Informationen über die Zusammengehörigkeit verschiedener Einzeldaten sind im Modell vorhanden.

Aufgebaut ist der Anfragebereich aus dem Basiselement und den darüber liegenden einzelnen Tabs, welche je ein separates Auswahlelement vom gleichen Typ sind. Diese können bei Fehleingabe wieder auf die Standardwerte zurückgesetzt oder ganz gelöscht werden. Mehrere Elemente (Tabs) werden bei der Anfrage als Verbunde (JOINs) über dieselbe Person aus bestimmten Tabellen interpretiert.

Neben den Hauptbereichen gibt es auf der Oberfläche auch noch weitere Elemente wie eine Menüleiste, eine Toolbar und eine Statusleiste. Die Toolbar enthält die wichtigsten Aktionen aus dem Menü zur bequemeren Bedienbarkeit. Jede Aktion wird deshalb abgegrenzt als einzelne Klasse, damit

sie von überall wiederverwendet werden kann und bei Änderungen nicht an verschiedenen Orten modifiziert werden muss. Dies entspricht dem MVC-Ansatz wie bereits im vorherigen Kapitel 5.1 diskutiert und trennt somit die Steuerung von der Präsentation.
Die Statusleiste enthält Informationen über die Verbindung mit der Datenbank. Dazu gehören der Verbindungsstatus (verbunden oder nicht verbunden), die Adresse, der Port sowie der Datenbankname.

Mit dem untenstehenden Klassenmodell (Abbildung 18) wird der Aufbau des GUI nochmals verdeutlicht. Die MainApplikation-Klasse erstellt eine MainGUI, wobei diese die Komponenten NavigationPaneComposite, ExportPluginComposite und möglicherweise weitere Komponenten im Einstellungsbereich lädt. Ausserdem werden die Tabs in der QueryComposite erstellt, wobei jedes Tab durch eine eigene DataItemComposite repräsentiert wird.

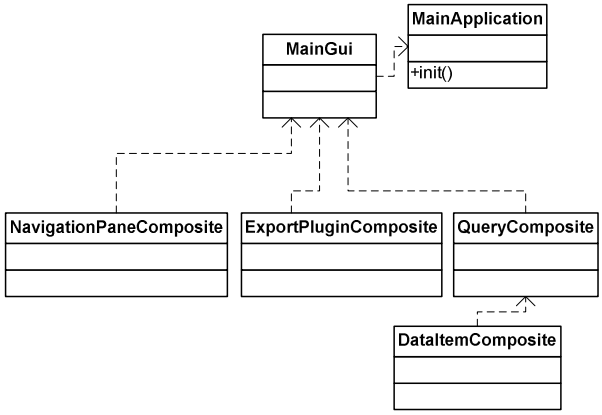

Abbildung 18: UML Klassendiagramm der GUI

Schliesslich wird über das Symbol die Export-Aktion ausgelöst, wodurch sich ein Speichern-unter-Dialog öffnet und der Benutzer einen gültigen Pfad und Dateinamen auswählen muss. Daraufhin wird überprüft, ob der Anwender alle notwendigen Informationen spezifiziert hat, um die Anfrage korrekt auszuführen. Ist dies nicht der Fall, wird der er dazu aufgefordert. Bei erfolgreicher Prüfung werden alle ausgewählten Parameter in die Anfragekonfiguration (Klasse QueryConfiguration) gespeichert und schlussendlich der Exportprozess ausgelöst.

5.2.3 Import des Metamodells

Damit dem Benutzer bei der Auswahl von Anfragedaten auch effektiv aktuelle Anfrageparameter zur Verfügung stehen, müssen die dafür notwendigen Metadaten bereits beim Programmstart vorliegen oder bei der Auswahl der Datenbank geladen werden. Da in einer ersten Umsetzungsphase lediglich eine Datenbankeinstellung gemacht werden kann, wird diese bei Programmstart standardmässig aus der in der Konfigurationsdatei definierten Datenbank geladen.
Für das Laden der Daten aus dem Metamodell ist eine eigene Klasse MetaModelReader verantwortlich. Sie bildet die benötigten Daten mittels Hibernate auf Java-Klassen ab. Um die Datenbank nicht übermässig zu belasten, werden immer nur diejenigen Daten geladen, die effektiv zur Auswahl der Anfrageparameter oder für den Export der Metadaten benötigt werden. Diese Daten

werden geladen und gepuffert. Werden noch nicht im Puffer vorhandene Daten benötigt, so werden diese geladen und im Puffer ergänzt. Dieser Ansatz entlastet die Datenbank insofern, als dass nicht schon zu Beginn die gesamte Datenmenge geladen werden muss, und dass bereits geladene Daten nicht bei der erneuten Verwendung wiederholt geladen werden müssen. Der Nachteil dieser Vorgehensweise ist, dass die Daten nicht jederzeit genau mit der Datenbank übereinstimmen müssen. Werden bereits geladene Daten während der Laufzeit des Sesam Export Managers in der Datenbank geändert, so wird dies nicht berücksichtigt und weiterhin auf die bereits in Puffer vorhandenen Daten zugegriffen. Da es jedoch wahrscheinlicher ist, dass Auswertungen über bereits vergangene Experimente gemacht werden sollen, sind diese meist schon vollständig in der Datenbank vorhanden und Änderungen nicht mehr zu erwarten. Ausserdem werden auch nur die Metamodelldaten geladen und nicht die Messungsdaten, die eher noch geändert werden könnten.

5.2.4 Export in Statistikformate

Der Datenexportprozess in den Sesam Export Manager wird durch eine Instanz der Klasse Query ausgeführt. Sie benötigt dafür einen gültigen Dateipfad, das für den Export zuständige Exportplugin sowie die durch den Benutzer spezifizierten Anfrageparameter. Die Klasse Query erstellt daraus ein Anfragestatement in SQL, öffnet eine Verbindung mit der Datenbank mittels JDBC (siehe Kapitel 5.3.3) und führt anschliessend das erstellte SQL-Statement aus. Als Ergebnis erhält sie ein Set von Datensätzen zurück. Dieses wird daraufhin dem gewählten Exportplugin für die Datenausgabe überreicht. Danach ist das Plugin für die Datenausgabe in die Datei zuständig. Die Verwendung eines Exportplugins lässt es offen, das Programm später ohne Änderungsaufwand um weitere Exportplugins für andere Exportformate zu erweitern. Umgesetzt wird dies in einem ersten Schritt durch die Verwendung einer abstrakten Klasse ExportPlugin. Dieser Aufbau wurde darum gewählt, weil er sehr schnell und einfach umgesetzt werden kann. Später könnte als komfortablere Lösung ein Ordner mit Exportplugins bereitgestellt werden, der zu Programmstart eingelesen würde.
Als Exportplugin wird in dieser Arbeit lediglich ein Plugin für SPSS erstellt. Aus den Anfrageinformationen und dem Ergebnisset erstellt es erst die Datei und schreibt den Header mit den entsprechenden Variablen und Skalentypen im SPSS-Format. Anschliessend werden über eine Schleife iterierend alle Datensätze aus dem Ergebnisset ausgegeben, wonach dann schliesslich die Ausgabe der Variablen- und Wertebeschriftungen erfolgt.
Mit dem SPSS Export Plugin werden alle Daten inklusive Metadaten in eine Datei im SPSS Syntax Format geschrieben. Eine weitere Möglichkeit wäre die Aufteilung der Daten und der Metadaten in zwei Dateien, wobei die eine die Form einer CSV-Datei hätte und die andere alle Metainformationen enthalten würde. So könnten die Daten auch beispielsweise mit Excel einfach betrachtet werden, da Excel CSV leicht lesen kann. Jedoch wurde das Plugin für SPSS geschrieben. Somit ist eine Datei handlicher als zwei Dateien, die auch immer im gleichen Ordner gespeichert werden müssen.

5.2.5 Generierung der SQL Anfrage

Der zur Anfrage benötigte SQL Befehl wird in mehreren Schritten anhand der Anfragekonfiguration QueryConfiguration generiert. Sie enthält alle ausgewählten Parameter. Nacheinander werden nun erst die SELECT-, die FROM- und zuletzt die WHERE-Klausel geschrieben.

Im der SELECT-Klausel werden erst für alle DataItemClasses die Schlüssel (ID) ausgelesen. Existieren zu den Schlüsseln in der Tabelle scie_data.additional_attribute_meta_data_now die zugehörigen Spaltenbeschriftungen, so werden diese selektiert. Sind keine vorhanden, so werden sie aus dem Datenbankschema ausgelesen, wobei einzelne Spalten wie tt_begin und tt_end standardmässig weggelassen werden. In der FROM-Klausel werden alle Tabellen data_item_<DataItemClass> sowie data_item_data_<DataItemClass>, und aus der DataItemClass, dem Expement, dem WorkflowNode, dem ProcessOutput sowie dem SubjectPriorityPath das DataItem (data_item_id) mit dem dazugehörige Probandenschlüssel (sid) ausgelesen. Mit der WHERE-Klausel werden alle resultierenden Datensätze auf Datensätze mit einheitlichen DataItemIds eingeschränkt. Die nachfolgende Abbildung 19 verdeutlicht den eben beschriebenen Ablauf:

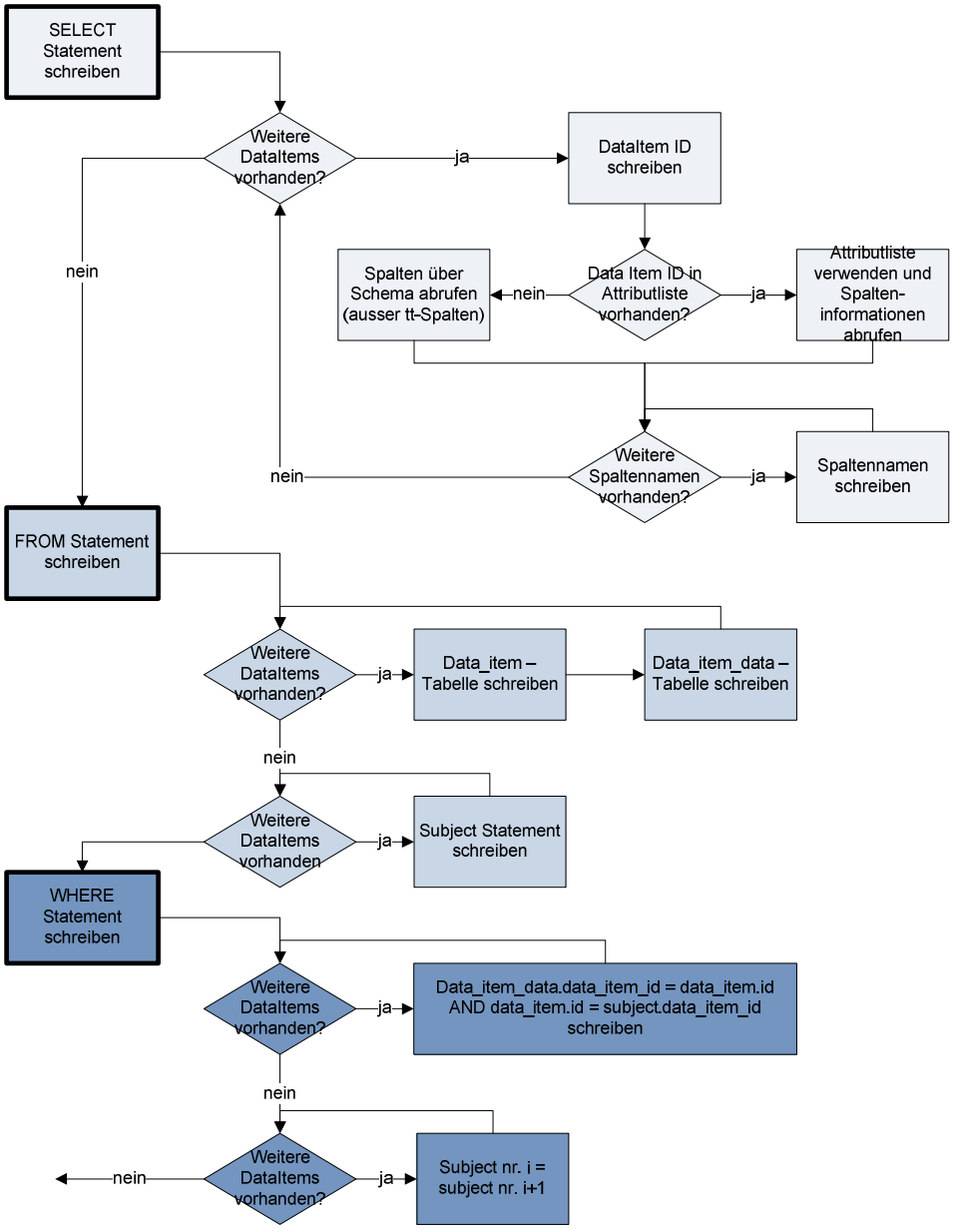

Abbildung 19: Generierung der SQL-Anfrage

5.2.6 Pseudonymisierung

Zur Pseudonymisierung der zu exportierenden Daten wird mittels einer Zufallsfunktion pro Person eine Zahl vom Typ Long als Identifikationsnummer generiert. Damit nicht mehrere Personen zufällig die gleiche ID erhalten, wird daneben eine Liste über bereits besetzte Zahlen und den dazugehörigen Personen geführt. Die folgende Abbildung 20 illustriert diesen Ablauf:

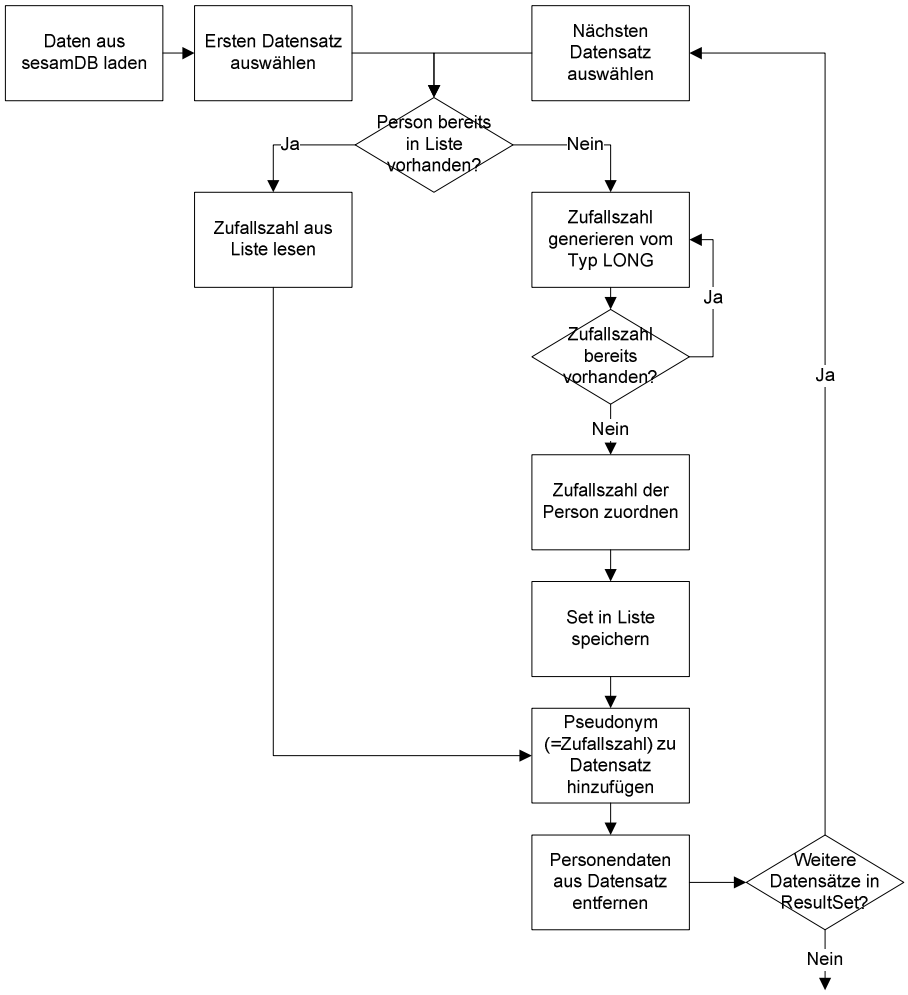

Abbildung 20: Prozess der Pseudonymisierung

Die Daten werden, wie oben illustriert, mit einer Anfrage aus der Datenbank geladen. Vom ResultSet wird anschliessend der erste Datensatz ausgewählt und geprüft, ob der Personenschlüssel bereits in der Zuordnungsliste geführt ist. Ist er es nicht, so wird eine Zufallszahl generiert. Damit diese eindeutig ist, darf sie noch nicht in der Zuordnungsliste vorhanden sein, ansonsten muss eine neue Zufallszahl generiert werden. Die Zufallszahl wird der Person zugeordnet und bildet somit das Pseudonym. Das Set Pseudonym und der Personenschlüssel werden in der Liste gespeichert, anschliessend wird das Pseudonym als weiteres Attribut auch noch in den Datensatz eingefügt. Existiert die Person schon in der Zuordnungsliste, so wird wieder dasselbe Pseudonym wie bereits zuvor verwendet. Schlussendlich werden alle personenbezogenen Informationen wie der Personenschlüssel gelöscht und der nächste Datensatz kann pseudonymisiert werden.
Diese Form der Pseudonymisierung wurde deshalb gewählt, weil sie einerseits einen ausreichend guten Datenschutz bietet (faktische Pseudonymität, vergleiche hierzu Kapitel 3.3), andererseits bleibt

es so möglich, in einem Export mehrere Datensätze über die gleiche Person miteinander zu vergleichen. Dies wäre bei einem Pseudonym pro Datensatz ohne Pseudonymliste nicht möglich. Hinzu kommt, dass dieser Pseudonymisierungsprozess einen relativ kleinen Implementierungsaufwand verursacht.

Das nächste Kapitel 5.3 beschäftigt sich mit den für die Entwicklung der Applikation verwendeten Hilfsmitteln.

5.3 verwendete Hilfsmittel

5.3.1 Java

Die Implementierung erfolgt in der objektorientierten Programmiersprache Java in der Version 1.5. Dies entspricht den Anforderungen an die Applikation.
Java-Programme werden in Bytecode übersetzt und dieser anschliessend auf einer virtuellen Maschine, der Java Virtual Machine, ausgeführt. Dadurch, dass die Java-VM für verschiedene Systeme wie Windows, Solaris oder Linux existiert, kann der Bytecode meist ohne Anpassung auf verschiedene Systeme übertragen werden und macht Java (zum grössten Teil) betriebssystemunabhängig.

5.3.2 SWT und JFace

Das Standard Widget Toolkit [SWT] ist ein vom Eclipse.org Konsortium entwickeltes Toolkit zur Gestaltung von Benutzeroberflächen in Java. Es bietet eine Alternative zu den häufig verwendeten AWT oder Swing APIs. Da in AWT komplexe Elemente wie beispielsweise Baumstrukturen fehlen, ist AWT oft nicht ausreichend. Swing verfügt zwar über viele verschiedene GUI-Elemente, zeichnet jedoch alle Befehle vollständig in Java und läuft deshalb sehr langsam. SWT legt eine zusätzliche Schicht über die Betriebsystem-API und benutzt dadurch die betriebssystemspezifischen GUI-Elemente. Aus diesem Grund benötigt SWT für jedes System eine eigene Bibliothek (wie beispielsweise swt-win32.jar). Durch die Verwendung von betriebssystemeigenen Widgets, kann ein SWT-Programm nicht mehr von nativen unterschieden werden (Warner & Harris, 2004).
Nachteilig wirken sich bei SWT der erhöhte Entwicklungsaufwand im Vergleich zu Swing sowie die fehlende Garbage-Unterstützung aus. Der belegte Speicher muss wieder freigegeben werden.
JFace ist ein GUI-Toolkit, das Hilfsklassen zur Erstellung von Benutzeroberflächen zur Verfügung stellt. Es nutzt dafür die von SWT gegebenen Widgets und bietet Klassen für häufige Programmieraufgaben, wie Viewer, Actions, Bild- und Schriftressourcen, Dialoge und Wizards sowie Formularassistenten (Warner & Harris, 2004).
Aufgrund der guten Performance und der relativ einfachen Implementierung von Benutzeroberflächen mit SWT und JFace wird das Sesam Export Manager GUI auch mithilfe dieser Toolkits implementiert. Ausserdem wird SWT und JFace bereits für andere sesam Applikationen verwendet, was den Einarbeitungsaufwand bei der Wartung der verschiedenen Applikationen reduziert.

5.3.3 JDBC

Die Java Database Connectivity [JDBC] API ist der Industriestandard für datenbankunabhängige Konnektivität zwischen Java und einer breiten Menge von Datenbanken. Wie Open Database Connectivity [ODBC] setzt sie auf dem X/OPEN SQL-Call-Level-Interface auf (Hartwig, 1998) und ist auch von der Funktionalität her vergleichbar mit ODBC oder Database Independent [DBI]. JDBC ist darum besorgt, die Datenbankverbindungen aufzubauen und zu verwalten sowie SQL-Anfragen an die Datenbank zu stellen und die Ergebnisse als Java-Objekt zurückzugeben. Für jedes Datenbanksystem (wie beispielsweise PostgreSQL, DB2 oder Oracle) wird ein spezifischer Treiber benötigt (Sun Microsystems, Inc., 2007). Die Verwendung von JDBC ist eine Anforderung an den Sesam Export Manager gemäss Kapitel 4.

5.3.4 Hibernate und andere ORM Frameworks

Hibernate ist ein von JBoss entwickeltes Framework für Java, das dem Programmierer ermöglicht, den Zustand eines Objekts in einer relationalen Datenbank zu speichern und aus den Datensätzen der Datenbank wiederum reale Objekte zu erzeugen. Dieses object-relational Mapping [ORM] vereinfacht die Entwicklung, da keine SQL-Anfragen mehr programmiert werden müssen (JBoss, 2007). Es lässt dem Benutzer jedoch offen, selbst Anfragen in Hibernate Query Language [HQL] oder in SQL zu schreiben.
Hibernate ist das Standard-Produkt für ORM in Java und besitzt eine grosse Community, welche das Framework pflegt und weiterentwickelt. Es gibt aber auch einige Alternativen wie beispielsweise Java Data Objects [JDO] von Apache (Apache, 2007), Enterprise Java Beans [EJB] aus dem Hause Sun (Sun Microsystems, Inc., 2007) oder Java Ultra-Lite Persistence [JULP] (Gurevich, 2007), die wie Hibernate eine relationale Datenbank auf Objekte zuordnet. Ein grosser Vorteil von Hibernate gegenüber vielen kommeriziellen JDO-Frameworks ist, dass es als Opensource und lizenzkostenfrei zur Verfügung steht. Ausserdem lässt sich sehr schnell damit entwickeln, da anders als in EJB keine Interfaces benötigt werden, sondern direkt mit einfachen Java-Klassen gearbeitet werden kann. Da Hibernate auch schon in weiteren sesamDB-Projekten eingesetzt wird und andere ORM-Lösungen keine signifikanten Verbesserungen darstellen, wird auch im Sesam Export Manager das bereits bekannte Framework Hibernate eingesetzt, was die Wartung der Software erleichtert. Vorteile gegenüber Hibernate bringt EJB vor allem in den Monitoring-Funktionen. Diese findet jedoch im Sesam Export Manager keine Verwendung.

5.4 Aufgetretene Probleme und Lösungsansätze

Die Implementierung des Sesam Export Managers erfolgte wann immer möglich gemäss den in Kapitel 5.1 ausgearbeiteten Spezifikationen. Als Hilfsmittel wurde eine Reihe von Programmen und Erweiterungen verwendet wie die Entwicklungsumgebung Eclipse Europe 3.3 in Kombination mit dem SVN-Plugin Subclipse von Tigris und dem Logging-Framework log4j der Apache Software Foundation.
Dieses Kapitel beschreibt einige ausgewählte Schwierigkeiten und Probleme, die während der Implementierung aufgetretenen sind, wie auch die dafür entwickelten Lösungsansätze.

5.4.1 Der Singleton-Ansatz

Bei Programmstart werden alle für die Exportanfrage benötigten Metamodelldaten unter Verwendung des MetaModelReaders importiert und als Java-Objekte gespeichert. Dieser Ladevorgang ist sehr ressourcenaufwendig und sollte nur einmal während des Programmstarts durchgeführt werden. Um zu verhindern, dass mehrmals dieselbe Klasse, in diesem Fall mehrmals ein MetaModelReader instanziiert wird, gibt es verschiedene Möglichkeiten: Die Klasse könnte statisch gemacht, als Singleton implementiert oder mit dem Factory Pattern entworfen werden.
Der Singleton ist ein Design Pattern und bewirkt, dass von der Singletonklasse nur ein einziges Objekt erzeugt werden kann. Er dient zur Erzeugung und Verwaltung von einzelnen Objekten einer Klasse und ermöglicht einen globalen Zugriff auf über eine statisch gebundene Methode, die im Sesam Export Manager getInstance() genannt wird. Diese Methode gibt (wie in Abbildung 21 ersichtlich) das private und statische Attribut ‚Instance' zurück. Der Klassenkonstruktor bleibt dabei private und somit gegen aussen nicht sichtbar (Krüger, 2004, S. 231).

```
private static MainApplication instance = null;

private MainApplication() {}

/**
 * make MainApplication singleton
 * @return
 */
public static MainApplication getInstance() {
      if (instance == null) {
            instance = new MainApplication();
      }
      return instance;
}
```

Abbildung 21: Singleton in Java

Die Verwendung von Klassen als Singletons ist ähnlich wie wenn die Klassen statisch wären. Allerdings bietet der Singleton-Ansatz den Vorteil, dass Änderungen später leichter möglich sind. Werden später mehrere Objekte derselben Klasse benötigt, kann entweder der Singletonaufbau modifiziert oder weggelassen werden. Nachteilig wirkt sich vor allem die schlechte Testbarkeit von Singletons aus. Werden mehrere Threads verwendet, so ist die Verwendung von Singletons nicht

mehr sinnvoll, da der Aufwand für die Sicherstellung von nur einer Instanz sehr gross ist. Ausserdem wird jeder Lesezugriff auch nach der Instanziierung über die getInstance()-Methode gemacht. Dies führt dazu, dass sich mehrere Threads gegenseitig blockieren können (Geary, 2003). Eine Alternative zu Singletons ist das Factory Pattern (Krüger, 2004, S. 234). Auch kann damit sichergestellt werden, dass nur eine Instanz gleichzeitig existiert. Allerdings ist der Implementierungsaufwand dafür bedeutend grösser.
Im Sesam Export Manager wird der Singleton-Ansatz angewendet, um zu verhindern, dass beispielsweise mehrere MetaModelReader oder mehrere Benutzeroberflächen gleichzeitig aktiv sind. Ausserdem ist so ein Zugriff auf die Instanz mittels der getInstance()-Methode auch ohne Referenz in der Klasse möglich.

5.4.2 Keine Kapselung des ResultSet möglich

Das während der Datenanfrage mittels JDBC genützte ResultSet hätte gemäss ursprünglicher Planung gekapselt und nach der Anfrage an die sesamDB zurück an die Controller-Klasse, die MainApplication, geliefert werden sollen. Von dort aus wäre sie dem zuständigen Exportplugin übergeben worden. Allerdings wird beim Zurückgeben des ResultSets an die MainApplication durch das Schliessen der Verbindung auch das ResultSet geschlossen, womit keine Datensätze mehr daraus abgerufen werden können.
Zur Lösung des Problems wird nun eine Ebene höher auf Stufe der Query-Klasse das ResultSet an das Exportplugin übergeben und die Verbindung erst dann geschlossen, wenn die Daten fertig exportiert sind. Dies hat den Vorteil, dass die Daten schnell wiedergegeben werden können. Allerdings schafft dieser Aufbau mehr Abhängigkeiten zwischen den Klassen.
Eine Alternative wäre die Rückgabe jeder einzelnen Reihe in eine Ergebnisklasse gewesen. Allerdings kann dies bei grossen Datenmengen rasch zu Performanceproblemen führen.

6 Evaluation der entwickelten Lösung

In diesem Kapitel wird der fertige Sesam Export Manager evaluiert. Dabei wird sowohl auf die Funktionalität der Anwendung wie auch auf die grafische Benutzeroberfläche eingegangen.

Ziel des Sesam Export Manager war es, Daten aus der sesamDB in ein Datenformat zu exportieren, dass von einem gängigen Statistikprogramm gelesen werden kann. Hierzu wurde als Exportformat das Syntaxformat von SPSS gewählt. Durch die Auswahl von Anfragekriterien in der grafischen Benutzeroberfläche kann der Anwender nun mit der vorliegenden Software Anfragen an die sesamDB stellen und erhält diese im entsprechenden Format inklusive deren Metadaten zurück. Zur Sicherstellung des Datenschutzes werden die Daten vor dem Export in das Statistikdateiformat pseudonymisiert. Diese in der aktuellen Version vorhandene Funktionalität entspricht der Basisanforderung für die Anwendung.

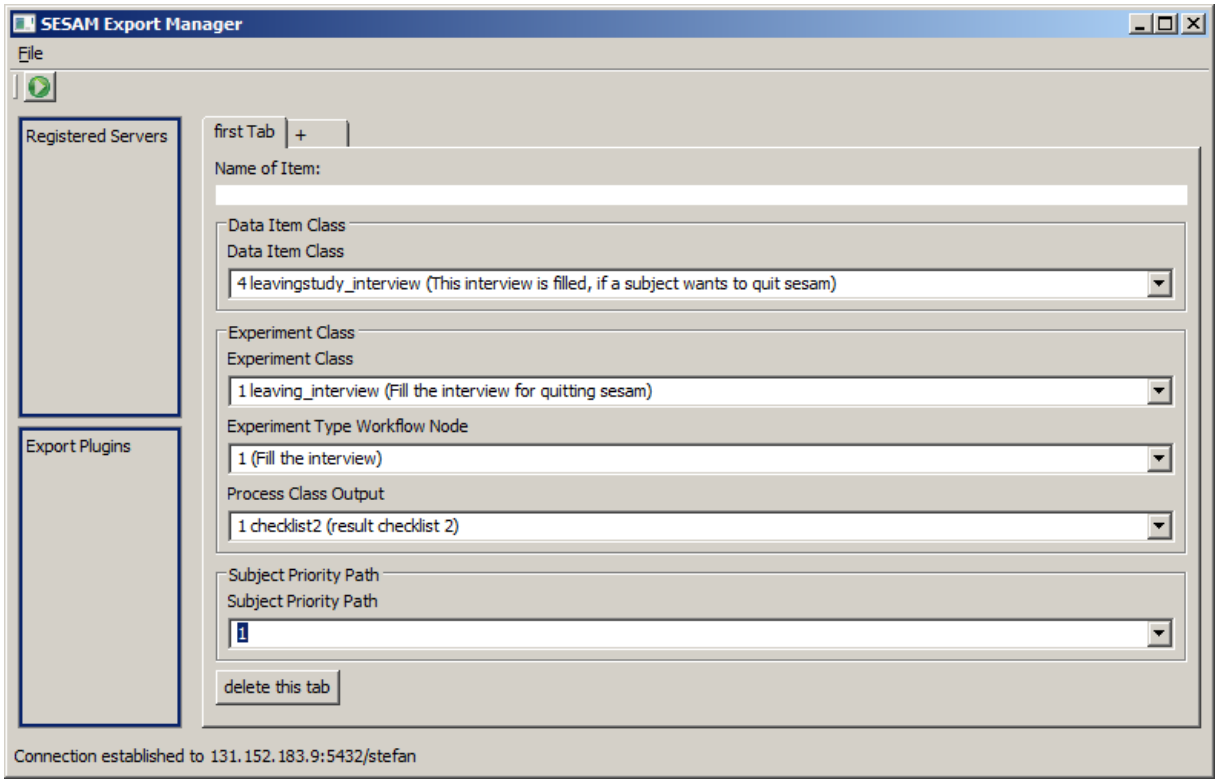

Abbildung 22: Screenshot vom Sesam Export Manager

Die obenstehende Abbildung 22 ist ein Screenshot aus der aktuellen Version von Sesam Export Manager. Die Umsetzung der GUI erfolgte weitgehend genauso, wie im Detailentwurf geplant. Sie repräsentiert die in der Datenbank zur Verfügung stehenden Datenobjektklassen in einer Auswahl (Combobox) und erlaubt es, diese für die Ausgabe einzelner Datenobjekte genau zu definieren. Dazu werden weitere Auswahlparameter wie die Experimentklasse, der Workflow-Knoten, die Prozessoutputklasse und die Rolle des Probanden spezifiziert. Durch das Hinzufügen weiterer Tabs, was durch Klicken auf das +-Tab ganz rechts geschieht, können weitere Datenobjektklassen der

Anfrage hinzugefügt werden. Diese können, durch klicken auf die Löschen-Schaltfläche, wieder entfernt werden.
Neben der Funktionalität und dem Erscheinungsbild der Applikation entspricht auch der Aufbau den Anforderungen aus Kapitel 4. Die Anwendung ist gemäss dem MVC-Ansatz in drei Teile gegliedert und lässt sich somit leicht mit weiterer Funktionalität erweitern. Zusätzliche Exportplugins für die Datenausgabe können durch Vererbung der Klasse ExportPlugin auf einfache Weise erstellt werden.

An dieser Stelle kann gesagt werden, dass die vorliegende Software den Hauptanforderungen an die Exportanwendung entspricht. Ihre Eignung in der Praxis müsste allerdings mithilfe ausführlicher Funktionstests (z.B. mit JUnit) sowie Benutzertests überprüft werden. Dies würde jedoch den Rahmen dieser Arbeit sprengen und ist Gegenstand weiterführender Arbeiten. Für mehr Benutzerkomfort und bessere Einsatzmöglichkeiten sollten vor der Einführung der Software die unter wichtige und hilfreiche Funktionalitäten eingestuften, noch fehlenden Anforderungen umgesetzt werden. Dazu gehören insbesondere die
Eingabe des Benutzerlogins über das GUI, das Aufrufen und Bearbeiten bereits durchgeführte Anfragen sowie der Zugriff auf in der Vergangenheit liegende Zustände der sesamDB. Für die Umsetzung der Auswahl von Server und Plugin über die GUI wurde zwar schon vorgesorgt und die notwendigen Komponenten bereitgestellt. Jedoch ist diese Funktionalität noch nicht implementiert, da erstens bisher nur ein Server und zweitens auch nur ein Plugin nötig waren. Die Auswahl der Server könnte in der GUI folgendermassen (siehe Abbildung 23) dargestellt werden:

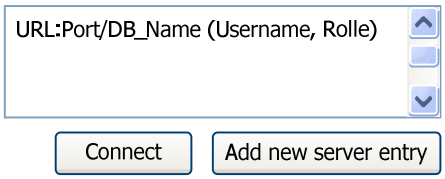

Abbildung 23: Entwurf für GUI-Element zur Serverauswahl

Zusätzlich zu den bereits definierten Anforderungen an die Weiterentwicklung könnte sich das Ersetzen der Combobox durch eine Suchmaske, wie in Abbildung 24 dargestellt, als praktikabel erweisen. Diese würde in der untenstehenden Liste alle Einträge enthalten, die bis anhin als Einträge in einer Combobox vorkamen. Zusätzlich würde während der Eingabe von Suchbegriffen die Liste nach relevanten Elementen gefiltert.

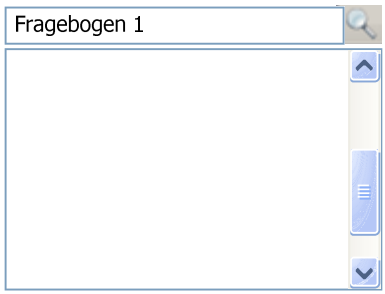

Abbildung 24: Entwurf für GUI-Element zur Suche nach Elementen aus der Datenbank

7 Zusammenfassung und Ausblick

Der Sesam Export Manager wurde als Exportanwendung für Datenbankinhalte der sesamDB entwickelt. Mit ihr steht dem sesam Projekt eine Applikation zur Verfügung, durch die Untersuchungsdaten in für Statistikprogramme lesbare Datenformate ausgegeben werden können. Das Besondere am Sesam Export Manager ist, dass auch Benutzer ohne Kenntnisse über Anfragesprachen wie SQL oder über die Datenbankstruktur in der Lage sind, sich die Daten aus der sesamDB zu laden, die sie für ihre Weiterverwendungszwecke benötigen. Dazu wurde eine komfortable grafische Benutzeroberfläche entwickelt, anhand der durch Auswählen von Experimenten, Prozessen und Datenobjekten automatisch die vom Benutzer erwarteten Daten zurückgegeben werden.
Diese Mechanik beruht auf einem dreistufigen Prozess, in dem im ersten Schritt während des Aufstartens der Applikation die verfügbaren Daten erst für den Anwender präsentiert werden. Aus diesen Daten wählt er die von ihm benötigten Datenobjekte aus und stösst den Export an. Im zweiten Schritt lädt der Sesam Export Manager die Daten aus der Datenbank in ein Ergebnisset und gibt diese schlussendlich im dritten Schritt über ein Exportplugin aus.
Zum Datenexport für Statistikprogramme liegt dem Export Manager ein Exportplugin für das SPS-Syntax Format für SPSS vor. Weil dieser Teil jedoch strikt vom Rest der Anwendung getrennt ist, können zur Erweiterung um neue Exportformate lediglich weitere Exportplugins geschrieben werden. Um die an der Studie beteiligten Personen zu schützen, werden die Daten beim Export jeweils pseudonymisiert. Dies geschieht durch Austauschen der Personenschlüssel durch Zufallszahlen.

Der modulare Aufbau des Programms ermöglicht es, Änderungen an der Applikation oder neue Funktionalitäten problemlos einzubauen. Als Ausblick auf Erweiterungen und Ergänzungen des Sesam Export Managers werden abschliessend folgende Funktionen als wichtig empfunden:

- **Filtern der Auswahl**
  Die Auswahl der Datenobjekte in der grafischen Oberfläche basieren zurzeit auf Comboboxen. Allerdings ist dies bei einer sehr grossen Auswahl nicht mehr praktikabel. Aus diesem Grund wäre es sinnvoll, die Comboboxen in der nächsten Version durch Listen mit einem dynamischen Suchfilter zu ersetzen.

- **Auswahl der Datenbank mit Benutzerangaben**
  In der jetzigen Version von Sesam Export Manager werden die Verbindungseinstellungen in der Konfigurationsdatei festgelegt. Sobald Zugriffsbeschränkungen für Benutzer oder Benutzergruppen eingeführt werden, benötigt die Anwendung eine Möglichkeit, Logindaten auch über die grafische Oberfläche einzugeben.

- **Auswahl des Exportplugins**
  Sobald der Applikation mehr als ein Exportplugin zur Verfügung steht, muss der Benutzer das gewünschte Datenformat angeben können. Dafür ist eine Auswahl des Plugins in der grafischen Benutzeroberfläche nötig.

- **Abrufen älterer Datenbankzustände**
  Damit Studien nachvollziehbar sind, müssen die Daten zu jedem Zeitpunkt im gleichen Umfang mit den gleichen Daten abrufbar sein. Dazu muss es möglich sein, frühere Zustände der Datenbank nach Datum und Uhrzeit auszuwählen zu können. Die Nachfrage nach einer solchen Funktionalität wird als sehr hoch eingeschätzt und wäre deshalb eine Anforderung für eine weitere Version.

- **Frühere Exporte öffnen und modifizieren**
  Damit für komplexere Exportanfragen nicht immer wieder die gleichen Angaben eingeben werden müssen, ist es sinnvoll, die früheren Exporte genauso oder modifiziert erneut auszuführen. Dazu müssen die Exporte zukünftig gespeichert und wieder aufgerufen werden können.

- **Verschlüsselte Verbindung**
  Bislang werden alle Datenverbindungen zwischen der Datenbank und der Exportapplikation ohne Verschlüsselung erstellt. Unter Umständen wäre es sinnvoll, hier eine Verschlüsselung der Verbindung einzuführen.

- **Benennung von Spaltenüberschriten**
  Damit es für den Benutzer einfacher ist, mit den Exportierten Daten zu arbeiten, müsste er Spaltenüberschriften selbst wählen können.

- **Auswahl/Ausblenden einzelner Spalten**
  In der aktuellen Version können immer nur alle Spalten einer Tabelle ausgewählt werden. Damit die Menge der exportierten Daten aber nicht zu schnell zu gross wird, müsste der Benutzer einzelne Spalten wählen bzw. abwählen können.

- **Filtern der Resultate**
  Eine weitere Möglichkeit die Menge der exportierten Daten so gering wie möglich zu halten und dem Benutzer die Applikation noch angenehmer zu gestalten, ist eine Filterung, beziehungsweise eine Einschränkung der Daten vor dem Export.

- **Vorschau der Resultate**
  Müssen die Daten immer vollständig exportiert werden, bevor sie vom Benutzer betrachtet werden können, kann dies für den Benutzer aber auch für die Datenbank viel Aufwand bedeuten. Eine Vorschau der ersten 100 Ergebnisse direkt in der Applikation würde die nächste Version des Sesam Export Managers erstens komfortabler machen und zweitens die Datenbank entlasten.

VII Literaturverzeichnis

Apache. (2007). *Apache Java Data Objects.* Abgerufen am 14. Dezember 2007 von http://db.apache.org/jdo/index.html

Berner, S., Glinz, M., Joos, S., Meier, S., Merlo-Schett, N., Ryser, J., et al. (2002). *Entwicklungsrichtlinien für Java-Software*, 3.0. (U. Zürich, Herausgeber) Abgerufen am 02. 12 2007 von http://www.ifi.unizh.ch/groupos/req/ftp/papers/JavaStyleguideV3.pdf

Bundesstatistikgesetz. (1. 4 2007). Abgerufen am 12. 12 2007 von http://www.admin.ch/ch/d/sr/4/431.01.de.pdf

Elmasri, R., & Navathe, S. B. (2005). *Grundlagen von Datenbanksystemen* (Grundstudium Ausg.). München: Pearson Studium.

Gaebel, W. (10. 10 2007). *Psychische Erkrankungen weiter auf dem Vormarsch*. Abgerufen am 2. 12 2007 von media.dgppn.de/mediadb/media/dgppn/pdf/presseinfo/2007/dgppn-pm07-17-welttag-seel-ges.pdf

Geary, D. (25. April 2003). *Simply Singleton.* Abgerufen am 10. Dezember 2007 von JavaWorld: http://www.javaworld.com/javaworld/jw-04-2003/jw-0425-designpatterns.html

Glavic, B., & Dittrich, K. (2006). *sesam: Ensuring Privacy for a Interdisciplinary Longitudinal Study.* Abgerufen am 02. 12 2007 von http://www.ifi.uzh.ch/dbtg/fileadmin/storage/Glavic/publications/06_GI_2006_workshop.pdf

Gurevich, L. (2007). *Java Ultra-Lite Persistence (JULP).* Abgerufen am 15. Dezember 2007 von http://julp.sourceforge.net/index.html

Hartwig, J. (1998). *Einführung in JDBC.* Abgerufen am 18. Dezember 2007 von http://web.f4.fhtw-berlin.de/hartwig/JDBC/jdbc.html

JBoss. (2007). Abgerufen am 20. Dezember 2007 von Hibernate: http://hibernate.org

Krüger, G. (2004). *Handbuch der Java-Programmierung.* München: Pearson Education.

Levesque, R. (2006). *SPSS Programming and Data Management* (3 Ausg.). Chicago, U.S.A.: SPSS Inc.

Middendorf, S., Singer, R., & Heid, J. (2002). *Programmierhandbuch und Referenz für die JavaTM-2-Plattform, Standard Edition.* Abgerufen am 11. 12 2007 von http://www.dpunkt.de/java/Programmieren_mit_Java/Oberflaechenprogrammierung/40.html

Murray, C., & Lopez, A. (1997). Alternative projections of mortality and disability by cause 1990-2020: Global Burden of Disease Study. *The Lancet* , S. 1498-1504 (349).

Pfitzmann, A. (2004). *Anonymity, Unobservability, Pseudonymity, and Identity Management – A Proposal for Terminology.* Abgerufen am 12. 12 2007 von http://dud.inf.tu-dresden.de/literatur/Anon_Terminology_v0.21.pdf

Pommerening, K. (17. 05 2000). *Pseudonyme – ein Kompromiß zwischen Anonymisierung und Personenbezug.* Abgerufen am 12. 12 2007 von http://www.staff.uni-mainz.de/pommeren/Artikel/pseudony.pdf

PostgreSQL Global Development Group. (2007). Abgerufen am 02. 12 2007 von PostgreSQL: http://www.postgresql.org/about/

Rost, M. (2003). Zur gesellschaftlichen Funktion von Anonymität. *Datenschutz und Sicherheit (DuD)* (27), S. 156-158.

SAS Institut. (2007). *SAS OnlineDoc 9.13.* Abgerufen am 8. Dezember 2007 von http://support.sas.com/onlinedoc/913/docMainpage.jsp

Schweizerischer Nationalfonds. (2005). *NFS SESAM - Schweizerische ätiologische Studie zur psychischen Gesundheit (SESAM)*. Abgerufen am 02. 12 2007 von http://www.snf.ch/D/forschung/Forschungsschwerpunkte/LaufendeNFS/Seiten/_xc_nfssesam.aspx

sesam Schweiz. (2007). *sesam Kernstudie: Besser verstehen – besser vorbeugen*. Abgerufen am 02. 12 2007 von http://www.sesamswiss.ch/sesam/ueber-sesam/kernstudie/nutzen/

sesam Schweiz. (2007). *sesam Teilstudien*. Abgerufen am 2. 12 2007 von sesam swiss: http://www.sesamswiss.ch/sesam/ueber-sesam/teilstudien/

Sun Microsystems, Inc. (2007). *Enterprise JavaBeans Technology.* Abgerufen am 15. Dezember 2007 von http://java.sun.com/products/ejb/

Sun Microsystems, Inc. (2007). *Sun Developer Network Industry Support.* Abgerufen am 20. Dezember 2007 von http://java.sun.com/products/jdbc/reference/industrysupport/index.html

UCLA Academic Technology Services. (2007). *SAS Learning Module: Inputting data into SAS.* Abgerufen am 8. Dezember 2007 von http://www.ats.ucla.edu/stat/SAS/modules/input.htm

Wahrig, G. (1997). *Deutsches Wörterbuch Wahrig.* Gütersloh: Bertelsmann Lexikon Verlag.

Warner, R., & Harris, R. (2004). *The Definitive Guide to SWT and JFace.* Berkeley, CA: Apress.