Fachhochschule Weihenstephan
University of Applied Sciences

Erstellung eines Datenbankschemas zur Verwaltung von Genexpressionsdaten

Diplomarbeit

Markus Alexander Obendorf

28. September 2005

Fachhochschule Weihenstephan
Fachbereich Biotechnologie
Prof. Dr. Bernhard Haubold
85354 Freising

Siemens AG
CT IC4, Neural Computation
Priv. Doz. Dr. Martin Stetter
Otto-Hahn-Ring 6, 81739 München

Eidesstattliche Erklärung

Gemäß §23 Abs. 6 der Prüfungsordnung

Ich erkläre hiermit an Eides statt, dass die vorliegende Arbeit von mir selbst und ohne fremde Hilfe verfasst und noch nicht anderweitig für Prüfungszwecke vorgelegt wurde.
Es wurden keine als die angegebenen Quellen oder Hilfsmittel benutzt. Wörtliche und sinngemäße Zitate sind als solche gekennzeichnet.

München, den 28. September 2005

Markus Alexander Obendorf

Danksagung

Im Rahmen dieser Diplomarbeit möchte ich mich bei einigen Personen bedanken. Mein Dank gilt der Siemens AG und Prof. Dr. Schürmann für die Bereitstellung des Themas. Ich bedanke mich bei Dr. Martin Stetter und Mathäus Dejori, unter dessen Leitungen ich diese Diplomarbeit verfasst habe. Ich möchte mich bei Prof. Dr. Bernhard Haubold bedanken, der mir seitens der Fachhochschule als Betreuer zur Seite stand. Abschließend gilt mein größter Dank meinen Eltern, die mir diese Ausbildung erst ermöglicht haben.

Inhaltsverzeichnis

Abbildungsverzeichnis

Abkürzungsverzeichnis

BCNF ..... Boyce-Codd Normal Form

cDNA ..... Complementary DNA

DNA ..... Desoxyribonucleic acid

FA ..... Funktionelle Abängigkeit

GEO ..... Gene Expression Omnibus

JDBC ..... Java Database Connectivity

JSP ..... JavaServer Pages

mRNA ..... Messenger RNA

NF ..... Normalform

OMG ..... Object Management Group

RNA ..... Ribonucleic acid

SMD ..... Stanford Microarray Database

SQL ..... Structured Query Language

XML ..... Extensible Markup Language

Kapitel 1

Einleitung

Die molekularbiologischen Möglichkeiten zur Entschlüsselung des genetischen Codes lebender Organismen haben die gesamte Biologie revolutioniert. Die komplette Analyse des menschlichen Erbmaterials in einem weltumspannenden Projekt (HUGO, Human Genome Project) [15] liefert das beeindruckendste Beispiel. Nachdem im Jahre 1953 die räumliche Struktur der DNA durch die Wissenschaftler James Watson und Francis Crick aufgeklärt wurde [3], ist bereits ein halbes Jahrhundert später das menschliche Erbgut sequenziert [13]. Aus der Sequenzierung der DNA hat sich in der Biologie einer neue Disziplin entwickelt: die Genomik.

Die Genomik beschäftigt sich mit dem Studium der Gene und deren Funktionen im Kontext ganzer Genome. Sie liefert detaillierte Kenntnisse über das Erbmaterial eines Zielorganismus und ermöglicht es den Wissenschaftlern, einzelne Gene zu identifizieren, die für bestimmte Proteine mit bestimmten Funktionen in einem Organismus verantwortlich sind. Die Herausforderung besteht darin, mit Hilfe geeigneter Technologien Funktionsweise und Zusammenspiel von Genen wissenschaftlich zu untersuchen. Eine dieser Technologien, die in den Laboren als Standard angesehen wird, ist die Microarraytechnologie.

1.1 Genexpression

Genexpression bezieht sich im Allgemeinen auf den Prozess, der die gespeicherte, kodierte Information eines Gens in ein Protein dekodiert. Die Genexpression unterteilt sich in Transkription und Translation. Die Transkription ist das Umschreiben des DNA-Strangs in eine komplementäre RNA-Sequenz mit Hilfe des Enzyms RNA-Polymerase. Die Translation ist der Vorgang, bei dem die Nukleotidsequenz der RNA in eine Aminosäuresequenz übersetzt wird [1].

In eukaryontischen Zellen findet die Transkription im Zellkern, die Translation im Cytosol statt. Das anfängliche RNA-Molekül, welches als Primärtranskript bezeichnet wird, enthält kodierende (Exons) und nichtkodierende Bereiche (Introns). In einem weiteren Vorgang, der als *Spleißen* bezeichnet wird, werden die Introns aus dem Transkript enzymatisch entfernt. Die resultierende mRNA wird vom Zellkern in das Cytoplasma transportiert, wo sie anschließend an den Ribosomen in das Protein übersetzt wird [1].

In prokaryontischen Zellen gestaltet sich die Synthese von mRNA wesentlich einfacher. Zum einen besitzen Prokaryonten keinen Zellkern, wodurch die mRNA- und Proteinsynthese in einem Kompartiment stattfinden, zum anderen ist das Spleißen nicht notwendig, da prokaryontische mRNA meistens keine Introns enthält [1].

1.2 Die Microarraytechnologie

Die mitte der neunziger Jahre an der Universität von Stanford entwickelte Microarraytechnologie ermöglicht die quantitative Messung der Genexpression in Zellen [10]. Das Prinzip dieser Technologie ist es, Biomoleküle mit bekannter Identität in Koordinatenform auf einen Objektträger zu immobilisieren, um anschließend eine Wechselwirkung mit markiertem Probenma-

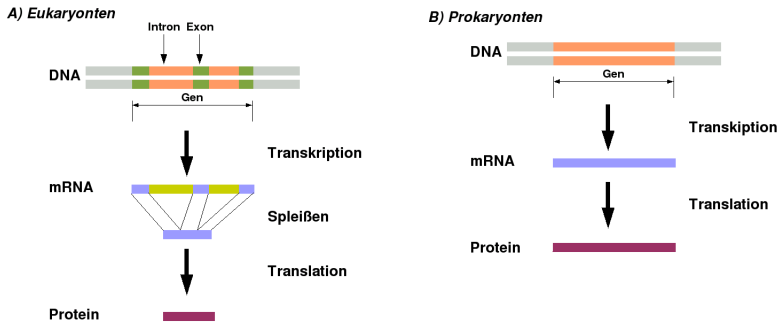

Abbildung 1.1: Genexpression in Pro- und Eukaryonten

terial hervorzurufen. Im Falle der Genexpressionsanalyse handelt es sich bei den Biomolekülen um DNA definierter Sequenz und bei dem Probenmaterial um revers transkribierter cDNA aus Zellen. Um die auf dem Trägersubstrat gebundenen Moleküle aus dem Probenmaterial lokalisieren zu können, erfolgt abschließend eine softwaregestützte Ermittlung [17].

1.2.1 Aufbau eines Microarrays

Grundbaustein eines Microarrays ist der aus Glas oder Nylon bestehende Objektträger. Seine Größe misst je nach Anwendung eine Breite von 75 mm und eine Tiefe von 25 mm. Auf ihm werden die Gene immobilisiert. Das Design eines Arrays ist der Struktur eines Koordinatensystems gleichzusetzen. Jeder Koordinatenpunkt wird als *Spot* bezeichnet, in dem sich mehrere tausend identische Kopien eines gleichen DNA-Segments befinden. Jeder Spot ist somit eindeutig über die Koordinatenposition (x, y) referenzierbar [17].

1.2.2 Funktionsweise eines Microarrays

Die Microarraytechnologie basiert auf dem Prinzip der Hybridisierung. Darunter ist zu verstehen, dass zueinander komplementäre DNA-Sequenzen über Wasserstoffbrückenbindungen miteinander sequenzspezifisch verbinden. Guanin und Cytosin binden über drei, Adenin und Tymin binden über zwei

Wasserstoffbrücken. Um die hybridisierten Sequenzen zu detektieren, werden die Probenmoleküle einem dem Arraytyp spezifischen Marker versetzt. Bindet nun das Probenmaterial an eine komplemantäre Nukleotidsequenz, ist die hybridisierte Stelle über den Marker lokalisierbar [17].

1.2.3 Arraytypen

Es gibt zwei Haupttypen in der Microarraytechnologie, die in Abhängigkeit der Sondenmoleküle, die sich auf dem Objektträger befinden, unterschieden werden können: cDNA- und Oligomicroarrays. Bevor das Probenmaterial auf das Array gegeben wird, muss es für den Versuch aufbereitet werden. Das bedeutet, dass die mRNA aus der Zelle extrahiert, mittels reverser Transkriptase in cDNA transkribiert und abschließend an einen spezifischen Marker gebunden wird (*engl. to label*) [17].

cDNA-Arrays

Auf einem cDNA-Array sind Nukleotidsequenzen einer Länge von bis zu 5000 Basen immobilisiert. Für diesen Arraytyp sind zwei Proben vorzubereiten. Eine Probe dient zur Kontrollmessung, die andere als Experimentmessung. Die zur Kontrollmessung gehörende cDNA wird mit einem grünen Fluoreszenzmarker gebunden (Cy3), jene für die Experimentmessung mit einem roten (Cy5). Beide Proben werden auf das gleiche Array gegeben. Es konkurrieren beide Biomolekültypen um eine Hybridisierungsstelle auf dem Array. Das Ergebnis der konkurrierenden Hybridisierung variiert in Abhängigkeit der cDNA-Konzentration von Kontroll- und Experimentprobe. Überwiegt die Anzahl an Hybridisierungen der cDNA aus der Kontrollzelle, so fluoresziert der Spot grünlich, im Falle der Experimentzelle rötlich. Ist die Konzentration an Transkripten aus beiden Zellen gleich, ist das ausgestrahlte Fluoreszenzlicht am Spot gelb. Erfolgte keine Hybridisierung, erscheint der Spot im Analysebild schwarz [17].

1.2.4 Oligoarrays

Auf diesen Arraytyp befinden sich DNA-Fragmente einer Länge von circa 25 Basenpaaren. Pro Spot sind 11 bis 20 identische Kopien platziert. Im Gegensatz zu cDNA-Arrays muss nur ein Probenansatz an Probenmaterial auf den Chip gegeben werden. Eine komplett hybridisierte Stelle wird als *perfect match* (abgekürzt: PM), eine nicht hybridisierte Stelle als *mismatch* (abgekürzt: MM) bezeichnet [17]. Bekanntestes Produkt eines Oligoarrys ist der *GeneChip* von Affymetrix.

1.3 Arrayherstellung

Für die Oligoarraytechnologie wird die Photolitographie angewendet. Diese wurde von Fodor *et al.* [17] entwickelt und von Affymetrix Inc. (Santa Clara, CA,USA) kommerzialisiert. In Abhängigkeit der Nukleotidsequenz werden Lochmasken mit Hilfe von Computeralogrithmen entworfen, die anschließend sequentiell auf die Arrayoberfläche platziert werden. Anschließend wird mittels UV-Licht die Microarrayoberfläche bestrahlt, wodurch nur jene Positionen chemisch aktiviert werden, welche das UV-Licht über die Lochmaske erreichen konnte. Abschließend kann das chemisch voraktivierte Nukleotid auf das Array gegeben werden, welches am Ende der Nukleotidsequenz anbindet. Dieser Vorgang wird solange wiederholt, bis die gewünschten Nukleotidsequenzen pro Spot erzielt wurden [17].

In der cDNA-Arraytechnologie werden bereits vorsynthetisierte DNA-Stücke an die Chipoberfläche gebunden. Die Methode wurde von Ed Southern [17] entwickelt und ist vom *Patrick O. Brown*-Labor an der Universität Stanford bekannt geworden. Dieser Arraytyp wurde in akademischen Labors deshalb favorisiert, da zur Herstellung keine Roboter notwendig waren. Die Kosten blieben somit pro durchgeführtem Experiment überschaubar [17].

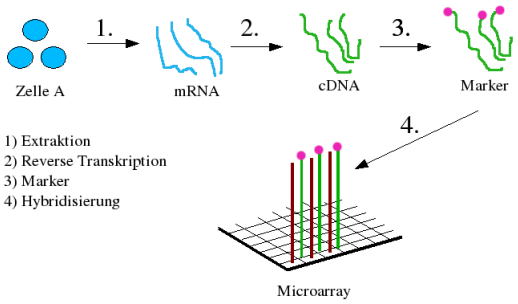

Abbildung 1.2: Arbeitsschritte im Umgang mit Microarrays

1.4 Motivation

Jedes Microarrayexperiment erzeugt eine Vielzahl an Daten. Den größten Anteil nehmen dabei die Scanwerte der einzelnen Gene ein. Diese Rohdaten sind für sich gesehen wertlos. Erst die biolologische Fragestellung, unter welcher das Experiment durchgeführt wurde, ermöglicht eine sinnvolle Interpretation und Auswertung der Daten. Im Internet stehen bereits Microarraydaten öffentlicher Datenbanken, wie SMD (*Stanford Microarray Database*) [16], GEO (*Gene Expression Omnibus*) [2] oder *ArrayExpress* [14] kostenlos zur Verfügung. Da jede dieser Datenbanken ihre proprietäre Datenstruktur besitzt, ist es wünschenswert, den Inhalt der Datenbanken zentral und standardisiert in eine Microarraydatenbank abzulegen, um zu den Genexpressionsdaten Modellberechnungen durchführen zu können.

1.5 Aufgabenstellung, Zielsetzung

Ziel dieser Diplomarbeit ist es, eine bereits bestehende MySQL Genomdatenbank um eine weitere Komponente zur Verwaltung und Speicherung von Microarraydatensätzen zu erweitern.

In einem ersten Schritt der Konzeptionierung sollen bekannte Datenbankschemata für Microarraydaten, wie beispielsweise MIAME (*Minimum information about a microarray experiment*) [12] geprüft und auf ihnen aufbauend ein eingenes Schema zur Speicherung und Verwaltung von Microarraydaten erstellt werden. Neben den eigentlichen numerischen Genexpressionsdaten sollen auch Informationen über Herkunft der Daten, Namen einzelner Experimente und etwaig duchgeführte Vorverarbeitungschritte gespeichert werden können.

In einem zweiten Schritt soll das entwickelte Datenbankmodell als relationale Datenbank in MySQL implementiert und in die bestehende Datenbank integriert werden. Desweiteren soll eine Benutzerschnittstelle in Java programmiert werden.

Im letzten Schritt soll die implementierte Datenbank mit Microarraydaten aufgefüllt werden. Neben bereits vorhandenen Daten sollen anhand einer Internetstudie öffentlich zugängliche Resourcen recherchiert und deren Daten in das System integriert werden.

Kapitel 2

Methoden

2.1 Materialien

Im Rahmen der Diplomarbeit standen folgende Arbeitsmaterialen zur Verfügung:

- Als Arbeitsplatzrechner eine Sun Solaris Workstation mit 256 MB Arbeitspeicher
- Für die Implementierung der Programmbibliotheken wurde das Java SDK 1.5 verwendet. Zum Ansprechen und Verbindungsaufbau der Datenbank ist die Open Source Version des JDBC-Treibers aus dem Internet bezogen worden.
- Die bereits vorhandene Datenbank war als MySQL der Version 4.0.23 vorinstalliert.
- Der Datenbankserver, auf dem die Microarraydaten abgelegt werden sollten, hatte als Betriebssystem Linux (Distributor: Fedora 2).
- Für die Entwicklung des Datenbankdesigns wurde das Genexpressionsmodell der OMG verwendet [5].
- Für die Entwicklung eines Webservers zur visuellen Darstellung der Datenbankinhalte wurde der Apache-Tomcat-Webserver bezogen.

2.2 Genexpressionsspezifikation der Object Management Group

Die Object Mangement Group, Inc. (OMG) wurde im Jahre 1989 gegründet und ist eine internationale Organisation, welche von über 600 Mitgliedern, überwiegend aus der Softwareenticklung, unterstützt wird. Ihr Aufgabengebiet fokusiert sich auf die Wiederverwendbarkeit und Portabiliät objektorientierter Software auf heterogenen Plattformen [5].
Die Genexpressionsspezifikation der OMG entstand in Zusammenarbeit mit Affymetrix, Inc. (Santa Clara, CA), Agilent Technologies, Inc. (Palo Alto, CA), Microarray Gene Expression Database (MGED) Society, National Center for Genome Resources (NCGR, Santa Fe, NM) und NetGenics, Inc. (Cleveland, OH). Grund für die Entwicklung dieser Spezifikation ist, Genexpressionsdaten verschiedener Technologien in einer Basissprache interpretieren zu können [5]. Sie ist in der Modellierungsprache UML (Unified Modeling Language) dargestellt. Die Unified Modeling Language ist eine formale Sprache zur Spezifikation, Visualisierung, Konstruktion und Dokumentation von Softwareprozessen und Geschäftsmodellen [4].
Die Spezifikation dient als Standard zur Darstellung von Genexpressionsdaten. Das daraus entwickelte Klassenmodell dient als Vorlage zur Entwicklung eines Datebankdesigns zur Abspeicherung von Genexpressionsexperimenten. Im folgenen soll nun das Vokabular, welches in der Genexpressionsspezifikation angewendet wird, näher dargestellt werden. Im Rahmen dieser Diplomarbeit werden nur jene Begriffe vorgestellt, die später in der Implementierung der Datenbank auch Anwendung finden.

2.2.1 Array

Die Klasse *Array* repräsentiert physikalisch gesehen das Microarray. Jede Instanz der Klasse *Array* besitzt das Attribut $arrayIdentifier$, über den

es referenziert werden kann. Unter dem Attribut $arryIdentifier$ ist zu verstehen, welchen Barcode der Objektträger hat, oder welchem Hersteller das Array zuzuordnen ist [5].

2.2.2 BioMaterial, MaterialType, BioSource

Auf jedes physikalische Array wird aufbereitetes Probenmaterial gegeben, welches im Modell als $BioMaterial$ bezeichnet wird. Die Klasse $BioMaterial$ stellt Substanzen wie Zellen, Gewebe, DNA oder bspw. Proteine dar. Damit die Bezeichnung für eine Instanz der Klasse $BioMaterial$ eindeutig ist, werden dieser Klasse die Ontologiereferenzen für Objekte der Klassen $MaterialType$ und $BioSource$ mitgegeben. Die Ontologie für die Klasse $MaterialType$ gibt das Vokabular für das verwendete biologische Material vor, wie bspw. DNA, RNA oder Lipide. Die Klasse $BioSource$ charakterisiert die Klasse $BioMaterial$. Darunter ist bspw. zu verstehen, welchem Zell- bzw. Gewebetypen das Probenmaterial zugeordnet werden kann [5].

2.2.3 BioAssay

In der Klasse $BioAssay$ werden Informationen dargestellt, die notwendig sind, wenn jeweils ein Objekt der Klasse $Array$ mit einem Objekt der Klasse $BioMaterial$ zusammengeführt wird. Im Laboralltag eentsprichtdas der Situation, wenn Probenmaterial auf den Genchip gegeben wird, mit der Absicht eine Hybridisierung zwischen Genen und extrahierter RNA hervozurufen. Die Klasse $BioAssay$ ist somit die Komposition aus den Klassen $Array$ und $BioMaterial$ [5].

2.2.4 ArrayDesign

Zu jeder Instanz der Klasse $Array$, welches in einem Objekt der Klasse $BioAssay$ verwendet wird, ist im Modell ein Objekt der Klasse $ArrayDesign$ zugeordnet. Unter der Klasse $ArrayDesign$ ist das Modell eines Microarrays

zu verstehen, das im Attribut *version* abgespeichert wird. Als zusätzliche Information wird die Spotanzahl im Attribut *numberOfFeatures* mitgeführt [5].

2.2.5 BioAssayData

Zu jedem im Labor durchgeführten Microarrayexperiment entsteht pro Spot ein numerischer Scanwert, der die Genaktivität der Zelle wiedergibt. Auf den Scanwerten erfolgen später die Analysemethoden. Die numerischen Werte werden in einem Objekt der Klasse *BioAssayData* abgespeichert, wofür das Eigenschaftsfeld *value* aufgenommen wurde [5].

2.2.6 CompositeSequence und Reporter

Die Klasse *CompositeSequence* speichert die zu einem Gen referenzierten Identifikationsschlüssel ab, hinter dem sich das auf dem Microarray immobilisierte DNA-Fragment verbirgt. Das Pendant zur Klasse *CompositeSequence* ist die Klasse *Reporter*. Sie repräsentiert den Identifikationsschlüssel für jene komplementäre Nukleotidsequenz, die an das DNA-Fragment hybridisiert [5].

2.2.7 BioSequence

Da sich jedes Objekt der Klasse *CompositeSequence* und *Reporter* indirekt aus einer definierten Nukleotidsequenz zusammensetzt, ist diese Information in der Klasse *BioSequence* dargestellt. Im Modell findet diese Klasse keine Anwendung, da die Nukleotidsequenz anderweitig referenziert werden kann [5].

2.2.8 ExperimentDesign

Die Klasse *ExperimentDesign* dient zur informativen Beschreibung eines Microarrayexperiments. Hier wird darauf eingegangen, unter welchen Aspekten

ein Versuch durchgeführt wurde. Darunter ist zu verstehen, ob bspw. ein Vergleich mit kranken und gesunden, oder behandelten und unbehandelten Patientenzellen gezogen wurde. Da für die Klasse *ExperimentDesign* definiertes Fachvokabular anzuwenden ist, ist eine Ontologie für die Beschreibung eines Objekts der Klasse *ExperimentDesign* separat mitzuführen [5].

2.2.9 ExperimentalFactor

Die Höhe des Expressionslevels eines Gens ist maßgeblich davon abhängig, welchen Versuchsbedingungen das Probenmaterial ausgesetzt war. Für die Interpretation der Scanwerte pro Gen sind diese Informationen von großer Bedeutung, da in einem darauffolgenden Schritt die Microarraydaten unter anderem in Abhängigkeit dieser Faktoren analysiert werden sollen. Diese Abhängigkeiten werden in der Klasse *ExperimentalFactor* dargestellt. Die Klasse *ExperimentalFactor* speichert beispielsweise die Tatsache, dass bei einem Microarrayexperiment die Glucosekonzentration variiert wurde, oder den Zeitverlauf, in der eine definierte Glucosekonzentration verabreicht wurde. Da es sich bei den Bezeichnungen um Fachvokabular handelt, ist der Klasse *ExperimentalFactor* das Attribut *ontology* vom Typ Ontology mitgegeben [5].

2.2.10 FactorValue, Measurement

Die Klasse *FactorValue* weist einem Objekt der Klasse *ExperimentalFactor* einen konkreten Wert zu. Ein Wert ist in diesem Zusammenhang bspw. die Höhe der Glucosekonzentration, oder in welchen Zeitintervallen ein Microarrayexperiment wiederholt wurde. Für die detaillierte Darstellung einer Instanz der Klasse *FactorValue* ist die Klasse *Measurement* vorhanden. Sie umfasst die Attribute *type*, das angibt, ob es sich dabei um eine Absolut- oder Differenzmessung handelt, *value*, in dem der Messwert für den Experimentparameter abgespeichert wird, und *kindCV*, welches den physikalischen

Parameter (Zeit, Temperatur, Masse, Volumen, ...) angibt [5].

2.2.11 Unit

Jede Instanz der Klasse *Measurement* benötigt einen Verweis darauf, in welcher Einheit die Werte eines Objekts der Klasse *FactorValue* vorliegen. Diese Aufgabe übernimmt im Modell die Klasse *Unit*. Sie besitzt eine Variable *unitName*, die die physikalische Einheit angibt [5].

2.2.12 Experiment

Ein *Experiment* ist sozusagen der Container aus dem sich ein Microarrayexperiment zu einer biologischen Fragestellung zusammensetzt. Nach dem Modell der OMG [5] ist das die Ansammlung an Instanzen der Klasse *BioAssay*. Für das Datenbankdesign werden in den Container die Klassen *BioAssayData*, *BioAssay*, *Description*, *ExperimentDesign* und *Contact* aufgenommen [5].

2.2.13 Audit

Die Dokumentierung eines Microarrayexperiments wird in der Klasse *Audit* mitgeführt. Für jedes Microarrayexperiment wird ein Datum festgehalten, an dem es erstellt bzw. Änderungen vorgenommen wurden. Desweiteren wird mit der Variablen *name* ermöglicht, dem Microarrayexperiment einen intuitiven Namen zu vergeben [5].

2.2.14 Contact

Die Klasse *Contact* soll verdeutlichen, welche Person bzw. Organisation das Microarrayexperiment durchgeführt hat. Eine Instanz der Klasse *Contact* enthält Datenfelder für Adresse, Telefonnummer, Email und Fax. Da ein Kontakt zum einen einer Person, zum anderen einem Institut zugeordnet

werden kann, ist die Klasse *Contact* auf die Klassen *Person* bzw. *Organization* erweiterbar [5].

2.2.15 Description

Die Klasse *Description* beschreibt eine Instanz der Klasse *Experiment* unter wissenschaftlichen Gesichtspunkten. Dazu verfügt die Klasse *Description* über ein Datenfeld namens *explanation*, in dem die Beschreibung zu einem Objekt der Klasse *Experiment* abgelegt werden soll [5].

2.2.16 BibliographicReference

Die Klasse *BibliographicReference* ist eine zu einem Microarrayexperiment veröffentliche wissenschaftliche Publikation. Ein Objekt dieser Klasse besteht somit aus dem Title, den Authoren und der Referenz auf das Journal. Zusätzlich wird das Datenfeld *abstract* hinzugefügt, in dem zu einer Veröffentlichung vorhandene Kurzzusammenfassung aufgenommen werden soll [5].

2.2.17 Anwendungsbeispiel

Zum besseren Verständnis des Klassenmodells sei ein konkretes Beispiel einer Genexpressionsstudie von *Saccharomyces cerevisiae* genommen, in der die Funktionalität des mRNA-bindenden Proteins *Puf3* untersucht wurde. Das Microarrayexperiment ist von *Gerber et al.* [9] publiziert worden, mit dem Ergebnis, dass das Protein fast ausschließlich an zytoplasmatischer mRNA bindet. Für das gesamte Experiment wurden der Wildtyp und eine *Puf3*-Mutante der Hefe verwendet, welche in einem Minimalmedium mit 3% Glycerol gezüchtet wurden. Der Versuchsansatz wurde dreimal unabhängig voneinander durchgeführt und ist aus der Microarraydatenbank von Stanford [16] bezogen worden. Modell und Realität sind in Abb. 2.1 dargestellt.

Die Pulikation von *Gerber et al.* ist im Modell einem Objekt der Klasse *BibliographicReference* zuzuweisen (Abb. 2.1, C). In einem Objekt der Klasse *Description* wird die Experimentbeschreibung abgespeichert, in der enthalten ist, dass es sich in diesem Microarrayexperiment um eine Genexepressionstudie von *Saccharomyces cerevisiae* handelt, dass Wiltdtyp und Mutante verwendet worden sind, und dass das Nährmedium mit Glycerol versehen wurde (Abb. 2.1, C). Da in diesem Versuch Wildtyp gegen Mutante gemessen worden ist, ist diese Information in einem Objekt der Klasse *ExperimentDesign* abzuspeichern (Abb. 2.1, A). Die Experimentparameter sind zum einem das im Nährmedium enthaltene Glycerol, zum anderen die n-te Wiederholung des Versuchs. Beide Faktoren werden jeweils in einem Objekt der Klasse *ExperimentalFactor* festgehalten (Abb. 2.1, B). In einem Objekt der Klasse *Measurement* wird im Attribut *type* abgespeichert, dass die Angabe der Glycerolkonzentration ein Relativwert ist, im Attribut *value* der angegebene Relativwert der Glycerolkonzentration (Abb. 2.1, B). Für die Klasse *Unit* sind aus dem Datensatz keine Informationen vorhanden. Der im Microarrayexperiment verwendete Organismus, *Saccharomyces cerevisiae*, ist einem Objekt der Klasse *BioSource*, die aus den Hefezellen zurücktranskribierte cDNA einem Objekt der Klasse *MaterialType* zuzuordnen. Beide Objekte zusammen assoziieren ein Objekt der Klasse *BioMaterial*. Für jedes verwendete Microarray ist ein Objekt der Klasse *Array* anzulegen. Die zu einem auf dem Microarray immobilisierten DNA-Fragmente referenzierten Identifikationsschlüssel sind jeweils in einem Objekt der Klasse *CompositeSequence*, der dazu referenzierte Schlüssel der komplementären Nukleotidsequenz in einem Objekt der Klasse *Reporter* abzuspeichern. Die Gesamtanzahl an Instanzen der Klassen *CompositeSequence* und *Reporter* sind in einem Objekt der Klasse *ArrayDesign* zusammenzuführen, welches dem zugehörigen Objekt der Klasse *Array* zuzuweisen ist. Da die Klasse *Array* und *BioMaterial* die Klasse *BioAssay* assoziieren, sind für dieses Mi-

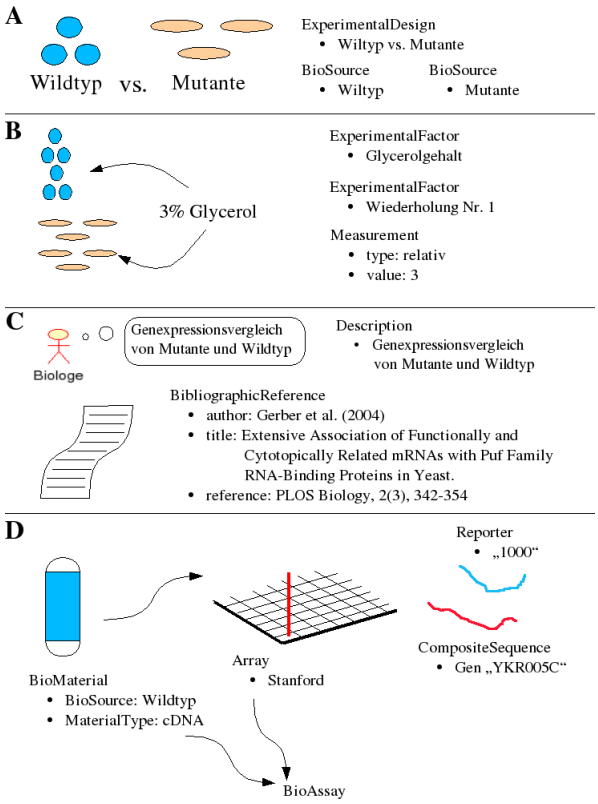

Abbildung 2.1: Modell und Realität

croarrayexperiment drei Instanzen der Klasse *BioAssay* zu erzeugen, da drei Proben angesetzt wurden (Abb. 2.1, D). Alle Informationen werden in einem Objekt der Klasse *Experiment* zusammengeführt, welches als Microarrayexperiment zu interpretieren ist.

Kapitel 3

Ergebnisse

Die Modellsprache der Genexpressionsspezifikation der OMG [5] dient nun als Grundlage zur Entwicklung eines Klassendiagramms, auf dem das zukünftige Datenbankdesign beruht. Aufbauend auf dem Klassendiagramm wurde ein Entity-Relationship-Modell entworfen, auf das die Beziehungen der einzelnen Klassen abgebildet werden. Das Entity-Relationship-Modell dient als Basis zur Implementierung der Tabellenstrukturen in MySQL. Damit Genexpressionsdatensätze in die Datenbank abgelegt werden können, wurde eine Applikation entwickelt, die aus dem Dateiinhalt ein Objekt der Klasse *Experiment* erzeugt und es anschließend in die Datenbank einspielt. Zur visuellen Darstellung der Datenbank wurde ein Webserver implementiert.

3.1 Klassendiagramm

Sinn und Zweck eines Klassendiagramms ist es, die statische Struktur eines Objektmodells zu beschreiben. In einem Klassendiagramm werden die Eigenschaften und gegenseitigen Beziehungen von Mengen gleichartiger Objekte aufgezeigt [11].

Alle Informationen, die bzgl. eines Microarrayexperiments abgespeichert werden sollen, werden in der Klasse *Experiment* (Abb. 3.1) abgelegt. Die

Klasse *Experiment* ist somit zentraler Kern des Modells. Ein Objekt der Klasse *Experiment* besteht aus den Instanzen der Klassen *Contact*, *BioAssay*, *BioAssayData*, *ExperimentDesign* und *Description*. Von der Klasse *Experiment* wird im Rahmen dieser Arbeit die Klasse *ArtificialExperiment* abgeleitet. Die Aufgabe dieser Klasse ist es, aus bereits bestehenden Objekten der Klasse *Experiment* oder *BioAssay* eine neue Instanz der Klasse *Experiment* erzeugen zu können.

Zwei weitere essentielle Klassen im Modell sind *BioAssay* und *BioAssayData*. Beide Klassen referenzieren ein Objekt der Klasse *Experiment*. Eine andere Möglichkeit wäre, dass die Klasse *BioAssayData* auf *BioAssay* referenziert. Nach der Genexpressionsspezifikation der OMG [5] sind beide Klassen separat gehalten, da die Klasse *BioAssay* im physikalischen und die Klasse *BioAssayData* im abstrakten Zusammenhang zu sehen sind. Physikalisch, da eine Instanz der Klasse *BioAssay* auf Instanzen der Klassen *Array* und *BioMaterial* verweist, Gegebenheiten, die real greifbar sind. Abstrakt, da in einem Objekt der Klasse *BioAssayData* Scanwerte abgelegt sind, auf dessen Interpretation Modellberechnungen erstellt werden. Die von der OMG angegebene Trennung dieser Klassen wird im Klassendiagramm übernommen. Die Klasse *BioAssayData* wird um das Attribut *format* ergänzt, worin jene Information abgespeichert werden soll, welcher mathematischen Nachbearbeitung die Scanwerte unterzogen worden sind.

Die Klasse *BioAssay* benötigt Referenzen auf Instanzen der Klassen *Array* und *BioMaterial*. Jede Instanz der Klasse *Array* verweist auf ein Objekt der Klasse *ArrayDesign*, diese wiederum auf eine Liste an Instanzen von *CompositeSequence* und *Reporter*.

Ausgehend von der Klasse *Experiment* ist eine Referenz auf die Klasse *ExperimentDesign* gelegt. Von der Modellvariablen *ExperimentDesign* wird auf die Klasse *ExperimentalFactor* verwiesen und von dort aus auf die Klasse *FactorValue*. Die Klasse *FactorValue* referenziert auf *Measurement*, und

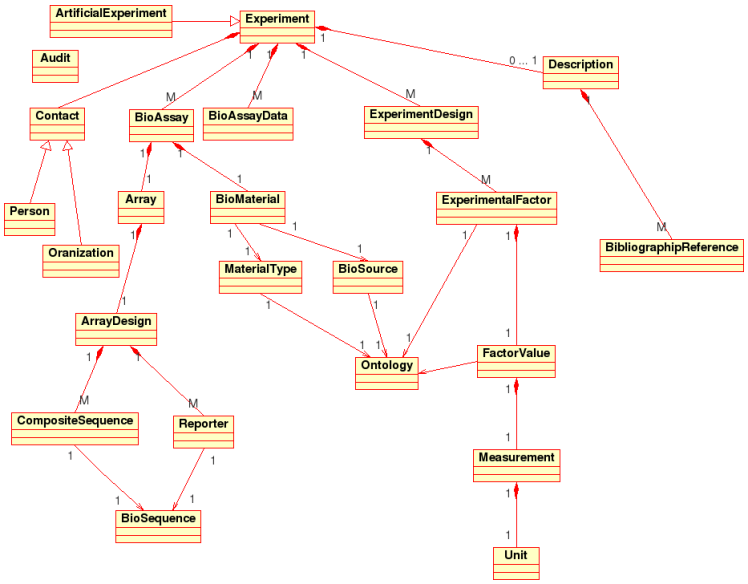

Abbildung 3.1: Klassendiagramm

diese auf *Unit*.

3.2 Entity-Relationship-Modell

Ein Entity-Relationship-Modell (abgekürzt E/R-Modell) beschreibt permanent gespeicherte Daten und ihre Beziehungen zueinander. Ergebnis eines E/R-Modells ist das E/R-Diagramm, das aus drei Grundbausteinen besteht: Entitäten, Attributen und Assoziationen. Ein Entitätstyp beschreibt Objekte, die anhand ihrer Eigenschaften unterschieden werden können. Entitätstypen besitzen gemeinsame Eigenschaften, die als Attribute bezeichnet werden. Beziehungen zwischen den Entitäten werden als Assoziationen bezeichnet. In einem Diagramm werden Entitäten als Rechtecke, Attribute als Ellipsen und Assoziationen über Rauten dargestellt [6].

3.2.1 Vom Klassendiagramm zum Entity-Relationship-Modell

Aus dem hergeleiteten Klassendiagramm können die Klassennamen direkt in Entitäten übernommen werden. Die Vorgehensweise zur Entwicklung des E/R-Modells bezieht sich auf den logischen Aufbau des Klassendiagramms aus Kap. 3.1 Abb. 3.1. Ausgehend von der Klasse *Experiment* werden alle Klassenbeziehungen durchlaufen und in ein E/R-Modell überführt.

Entität: Experiment

Die Entität *Experiment* (Abb. 3.2) hat Assoziationen zu den Entitäten *BioAssay*, *BioAssayData*, *Description* und *ExperimentDesign*. Ein Objekt vom Typ *Experiment* besitzt mindestens ein Objekt vom Typ *BioAssay*, *BioAssayData* und *ExperimentDesign*. Die Komplexität ist einfach-komplex (1:M). Bei der Entität *Description* ist es prinzipiell möglich, dass es zu einem Microarrayexperiment keine, aber maximal eine Beschreibung existiert.

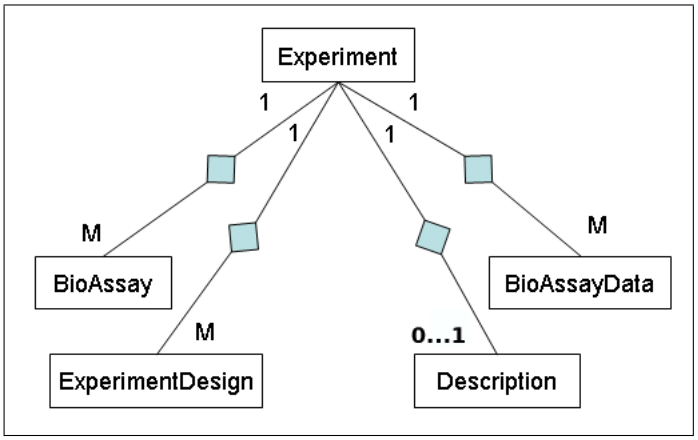

Abbildung 3.2: E/R-Modell für Entität *Experiment*

Entitäten: BioAssay, BioAssayData

Nach dem Klassendiagramm (Kap. 3.1, Abb. 3.1) sind die Klassen *BioAssay* und *BioAssayData* voneinander getrennt. Für das Datenbankdesign kann im

E/R-Modell diese Trennung nicht mehr vorgenommen werden. Die Entitäten *BioAssay* und *BioAssayData* (Abb. 3.3) stehen über eine Assoziation in Verbindung. Da die Scanwerte zu einem Gen in verschiedenen Formaten vorliegen können, ist es möglich, dass zu einem Objekt der Entität *BioAssay* mehrere Objekte der Entität *BioAssayData* vorliegen, mindestens jedoch eines. Die sich daraus ergebende Kardinalität zwischen beiden Entitäten ist einfach-komplex.

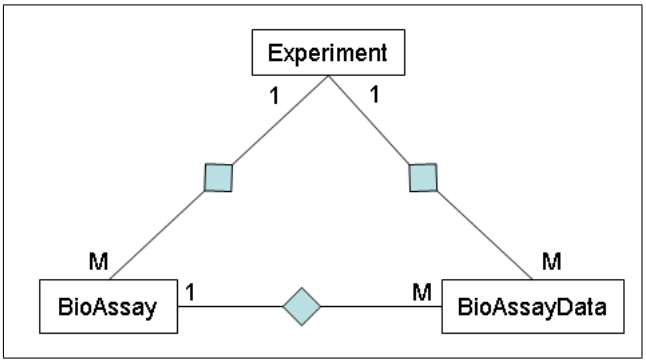

Abbildung 3.3: E/R-Modell für Entitäten *BioAssay, BioAssayData*

Entitäten: BioAssay, Array, BioMaterial

Jede Instanz der Entität *BioAssay* verfügt über genau eine Beziehung zu den Entitätstypen *Array* und *BioMaterial* (Abb. 3.4). Da auf ein Microarray nur eine Art an Probenmaterial gegeben wird, ist die Beziehung zwischen den Entitäten *Array* und *BioMaterial* ebenfalls einfach-einfach.

Entitäten: Array, ArrayDesign, CompositeSequence, Reporter, BioSequence

Da sich jede Instanz der Klasse *ArrayDesign* auf ein Objekt der Klasse *Array* bezieht, ergibt sich zwischen ihnen eine einfach-einfach Beziehung. Die Kardinalität zwischen den Entitäten *ArrayDesign* und *CompositeSequence*

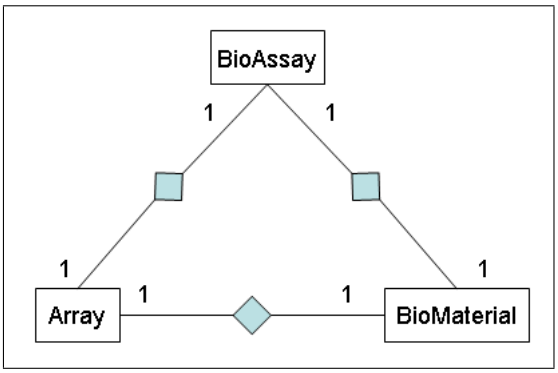

Abbildung 3.4: E/R-Modell für Entitäten *Array, BioMaterial*

bzw. *Reporter* ist einfach-komplex, jene zwischen *CompositeSequence* und *Reporter* einfach-einfach (Abb. 3.5).

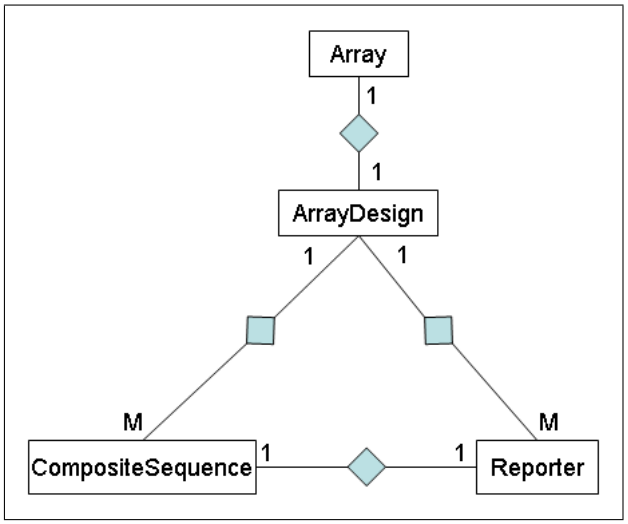

Abbildung 3.5: E/R-Modell für Entitäten *CompositeSequence, Reporter*

Entitäten: Experiment, ExperimentDesign, ExperimentalFactor, FactorValue, Measurement

Die Entität *Experiment* verweist auf eine oder mehrere Entitäten des Typs *ExperimentDesign* und dieses wiederum auf eine oder mehrere Entitäten des

Typs *ExperimentalFactor* (Abb. 3.6). In einer einfach-einfach Beziehungskette geht es ausgehend von der Entität *FactorValue* über die Entität *Measurement* zur Entität *Unit*.

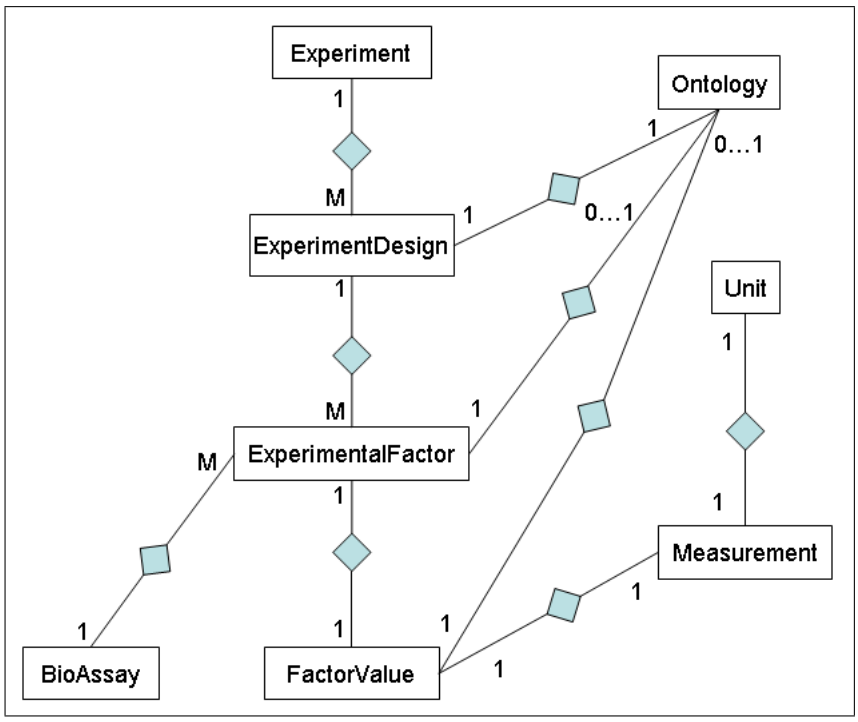

Abbildung 3.6: E/R-Modell für Entität *ExperimentDesign*

Entitäten: Description, BibliographicReference

Im E/R-Modell muss von der Beziehungskette aus dem Klassendiagramm (Kap. 3.1, Abb. 3.1) *Experiment–Description–BibliographicReference* abgewichen werden. Es ist denkbar, dass zu einem Microarrayexperiment keine Beschreibung vorhanden ist, dennnoch aber eine oder mehrere Veröffentlichungen. Die Assoziation *Description–BibliographicReference* wird deshalb geändert auf *Experiment–BibliographicReference*, was ermöglicht, dass im Falle der Nichtexistenz einer Beschreibung zu einem Experiment dennoch eine oder mehrere Publikationen referenziert werden können (Abb. 3.7).

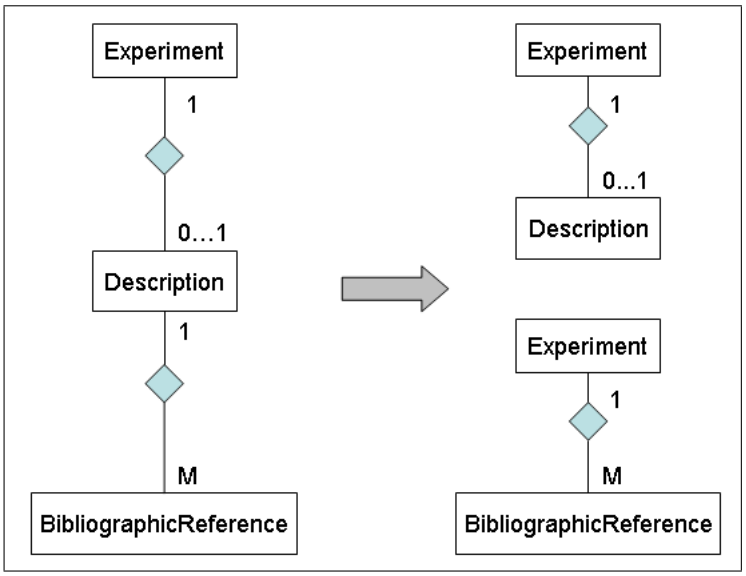

Abbildung 3.7: E/R-Modell für Entität *Description, BibliographicReference*

Entitäten: Audit, Contact

In der Entität *Audit* (Abb. 3.8) können eine oder mehrere Objekte der Entitätstypen *Experiment* aufgelistet sein. Die Beziehung zwischen ihnen ist einfach-komplex gehalten. Zu jeder Entität *Experiment* ist ein Objekt der Entität *Contact* vorhanden.

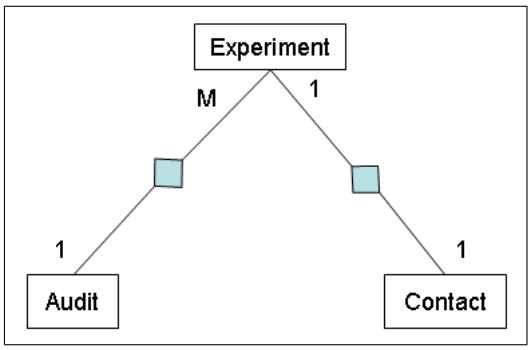

Abbildung 3.8: E/R-Modell für Entität *Audit, Contact*

Abschließend sind in Abb. 3.9 alle Entitäten und ihre Beziehungen zu-

einander aufgeführt.

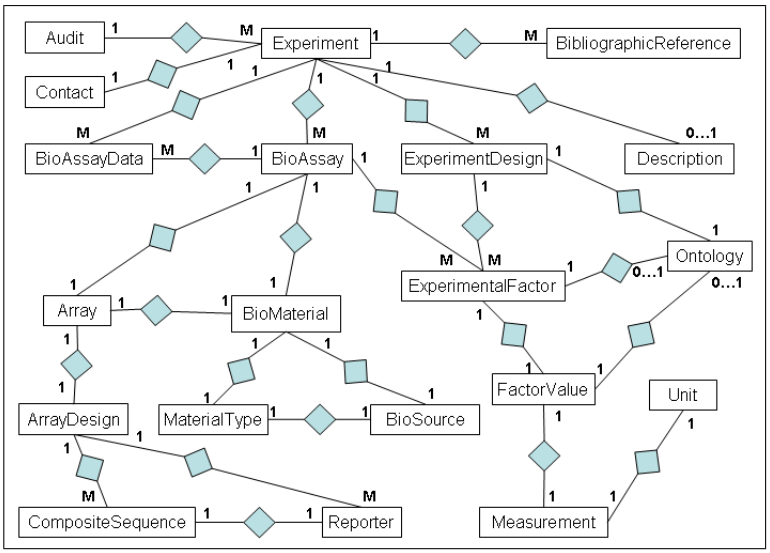

Abbildung 3.9: E/R-Modell im Gesamtüberblick

3.3 Funktionelle Abhängigkeiten und Normalformen

Aus den funktionellen Abhängigkeiten, die sich aus dem Klassenmodell erschließen lassen, soll nun anhand von Anwendungsfällen die optimalen Tabellenformen für das Datenbankdesign hergeleitet werden. Das Ziel ist, jede hergeleitete Tabelle in Boyce-Codd Normalform (BCNF) zu überführen. Dazu wird die Dekomposition in BCNF angewendet. Das bedeutet, dass pro Attribut bzw. Attributkombination die Abschlussmenge berechnet wird, bis eine nichttriviale funktionelle Abhängigkeit gefunden werden kann, die die BCNF verletzt.

3.4 Definition: Funktionelle Abhängigkeit (FA)

Sei R eine Relation über die Attributsmengen X und Y. Das Attribut Y ist funktional abhängig vom Attribut X, $X \rightarrow Y$, genau dann, wenn zu jedem Wert X genau ein Wert von Y zugwiesen werden kann [7].

3.5 Definition: 1. Normalform

Eine Relation R ist in 1. Normalform (1.NF) genau dann, wenn jedes Tupel dieser Relation genau einen Wert pro Attribut enthält. In relationalen Datenbanken wie MySQL sind alle Tabellen in 1. NF, da keine Wiederholungsgruppen möglich sind [7].

3.6 Definition: 2. Normalform

Eine Relation R ist in 2. Normalform (2. NF) genau dann, wenn sie in 1. NF ist und gleichzeitig jedes Nichtschlüsselattribut vollständig vom Schlüssel abhängig ist [7].

3.7 Definition: 3. Normalform

Eine Relation R ist in 3. Normalform (3. NF) genau dann, wenn sie in 2. NF ist und jedes Nichtschlüsselattribut nichttransitiv vom Schlüssel abhängig ist [7].

3.7.1 Definition: BCNF

Eine Relation R ist in Boyce-Codd Normalform genau dann, wenn jede nichttriviale, linksirreduzible funktionelle Abängikeit einen Schlüsselkanditaten als Determinante enthält. Eine Tabelle die in BCNF ist, ist ebenfalls in 3. NF [7].

3.7.2 Definition: Abschlussmenge

Die Menge aller funktionellen Abhängigkeiten, die durch eine gegebene Menge S an FA's impliziert werden kann, wird als Abschlussmenge von S bezeichnet und mit S^+ abgekürzt [7].

3.7.3 BCNF-Verletzung

Aus der Definition der BCNF (Kapitel 3.7.1) und Abschlussmenge (Kapitel 3.7.2) kann in Bezug auf die BCNF-Verletzung geschlossen werden, dass nur dann eine Verletzung vorliegt, wenn in der Abschlussmenge nicht alle Attribute aus der gegebenen Relation R enthalten sind.

3.7.4 Anwendungsfälle

Instanzen der Entität Experiment und Audit abspeichern

Zu einer gegebenen Instanz der Klasse *Experiment* wird eine Instanz der Klasse *Audit* mitgeführt, die in einer Tabelle separat abgespeichert werden soll. Ist eine Instanz der Entität *Experiment* bekannt, so ist auch deren Instanz der Klasse *Audit* erschließbar. Daraus ergibt sich die funktionelle Abhängigkeit:

$$Experiment \rightarrow Audit \;\Big|\; A \rightarrow B$$

Die Abschlussmengen A^+, B^+ und $\{AB\}^+$ dieser Relation liefern keine Verletzung der BCNF, da $\{AB\}^+$ und B^+ trivial sind und mit $A^+=\{AB\}$ zugleich ein Superschlüssel und Schlüssel gefunden werden konnte. In der Datenbank wird diese Relation mit dem Tabellennamen *audit* bezeichnet. Das Attribut A der Relation R wird mit *exp_ID* bezeichnet und ist Schlüsselkandidat. Da das Attribut B eine Instanz der Klasse *Audit* darstellt, kann es noch nicht in die Tabelle aufgenommen werden. Die Lösung liegt darin, dass jedes Attribut der Klasse *Audit* in eine Spalte überführt wird. Da in der Klasse *Audit* keine Attribute vom Typ Array abgelegt sind, tritt in Bezug

auf die Relation R(A, B) keine Verletzung der Boyce-Codd Normalform auf, da die Attribute *date*, *name*, *gsname* und *updated* der Klasse *Audit* weiterhin funktional abhängig zu einer Instanz der Klasse Experiment sind, und damit vom Schlüssel *exp_ID*.

Zu einer Instanz der Entität Experiment dessen Instanzen BioAssay, und zu einer Instanz BioAssay dessen Instanzen BioAssayData abspeichern

Da die Beziehungen zwischen den Entitäten *Experiment* und *BioAssay*, *BioAssay* und *BioAssayData* jeweils einfach-komplex sind (Kap. 3.2.1, Abb. 3.2), geht die Eindeutigkeit der Beziehung nur vom komplexen Anteil der funktionellen Abhängigkeit aus, dem Dependenten. Somit ist eine Instanz der Entität *Experiment* funktional abhängig von dessen Entitäten *BioAssay*, und diese der zugehörigen Entität *BioAssayData*. Das Gesetz der Transitivität ergibt die funktionelle Abhängigkeit der Entität *Experiment* als Dependenten und *BioAssayData* als Determinante.

$$\begin{array}{l|l} BioAssayData \rightarrow BioAssay & A \rightarrow B \\ BioAssay \rightarrow Experiment & B \rightarrow C \\ BioAssayData \rightarrow Experiment & A \rightarrow C \end{array}$$

Die Berechnung der Abschlussmengen (Abb. 3.10) der Attribute aus der gegebenen Relation R (A, B, C) ergibt mit B^+={BC} eine Verletzung der BCNF. Die Relation R wird mit $B \rightarrow C$ in die Relationen R_1 (A, B) und R_2 (B, C) aufgetrennt. Die Relationen R_1 und R_2 sind bereits in BCNF. Die Relation R_1 mit dem Schlüssel A wird in der Datenbank als *bioAssayData_bioAssay* und R_2 mit Schlüssel B als *assay_data_experiment* referenziert.

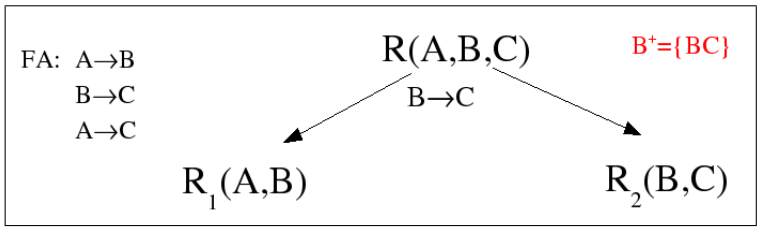

Abbildung 3.10: BCNF-Zerlegung für R(A,B,C)

Zu einer Instanz der Entität BioAssay die Instanzen der Entitäten Array, BioMaterial, Array, ArrayDesign, CompositeSequence und Reporter abspeichern

Nach dem E/R-Modell ergeben sich, unter der Voraussetzung, dass eine Instanz der Klasse *BioAssay* zu einem Objekt des Typs *Experiment* eindeutig zuzuordnen ist, folgende funktionelle Abhängigkeiten. Mit diesem Anwendungsfall ist dieRelationn R (A, B, C, D, E, F, G) gegeben.

$BioAssay \rightarrow Array$	$A \rightarrow B$
$BioAssay \rightarrow ArrayDesign$	$A \rightarrow C$
$BioAssay \rightarrow BioSource$	$A \rightarrow D$
$BioAssay \rightarrow MaterialType$	$A \rightarrow E$
$ArrayDesign \rightarrow Array$	$C \rightarrow B$
$Reporter \rightarrow CompositeSequence$	$F \rightarrow G$

Ist A bekannt, so ist aufgrund der einfach-einfach Beziehung von A zu B, B eindeutig zuzuordnen. Funktional abhängig von A ist ebenfalls D und E. Wenn C bekannt ist, kann auf B geschlossen werden und F impliziert G.

Die BCNF-Zerlegung der Relation R(A, B, C, D, E, F, G) (Abb. 3.11) ergibt mit der BCNF-Verletzung $A \rightarrow BCDE$ die Relationen R_1 (A, B, C, D, E) und R_2(A, F, G). R_1 ist in der NF von Boyce-Codd und darf gemäß dem Theorem von Heath weiter aufgeteilt werden. Das Aufspalten der Relation R_1 ist aus datenbanktechnischer Sicht nicht notwendig, orientiert sich

aber näher am Klassendiagramm (Kap. 3.1, Abb.3.1). Die Aufteilung wird mit dem Schlüssel A in die Relation R_3(A, B, C) und R_4(A, D, E) gelenkt. Relation R_2 enthält mit $F \rightarrow G$ eine BCNF-Verletzung. Die Zerlegung ergibt R_5(F, G) und R_6(A, F).

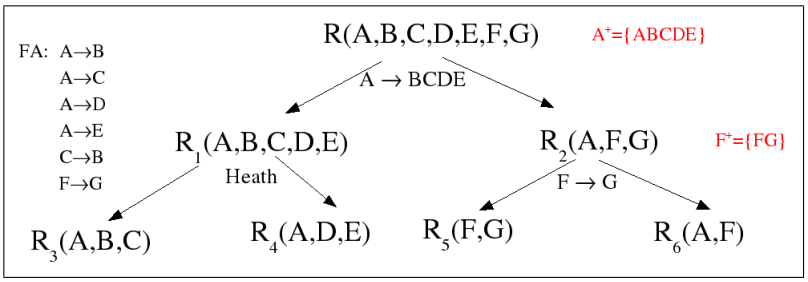

Abbildung 3.11: BCNF-Zerlegung

Die Interpretation der Relation R_6 ergibt, dass sich zu jeder Instanz der Klasse *BioAssay* die in die Datenbank eingespielt werden soll, die gesamte Anzahl an Reporterinstanzen zu einem Microarray abgespeichert werden. Diese Tatsache kann in der Hinsicht nicht akzeptiert werden, da es zum einen wünschenswert ist, die Reporterinstanzen der Klasse *ArrayDesign* zuordnen zu können, wie es im Klassendiagramm (Kap. 3.1, Abb. 3.1) vorgesehen ist. Zum anderen entsteht in der Relation R_6 eine Redundanz, da zu jeder Instanz der Klasse *BioAssay* die Reporterinstanzen direkt abgespeichert werden. Beinhaltet ein Objekt der Klasse *Experiment* n Objekte der Klasse *BioAssay*, und jede Instanz davon das gleiche physikalische Microarray, so wird n-mal die gleiche Anzahl an Reporterinstanzen in der Tabelle R_6 abgelegt. Die Tabellenzerlegung (Abb. 3.12) wird so abgeändert, dass nur von den Attributen A, B, C, D und E ausgegangen wird. Für die Attribute F und G wird separat eine Tabelle geführt.

Dazu werden die Hilfsrelationen R_α(C, X) und R_β(X, G, H) mit $X \rightarrow GH$ angelegt. In der Relation R_β ist mit dem Attribut X ein Identifikationsschlüssel zu G und H gegeben. Relation R_α ist nur in 1. NF, da aufgrund

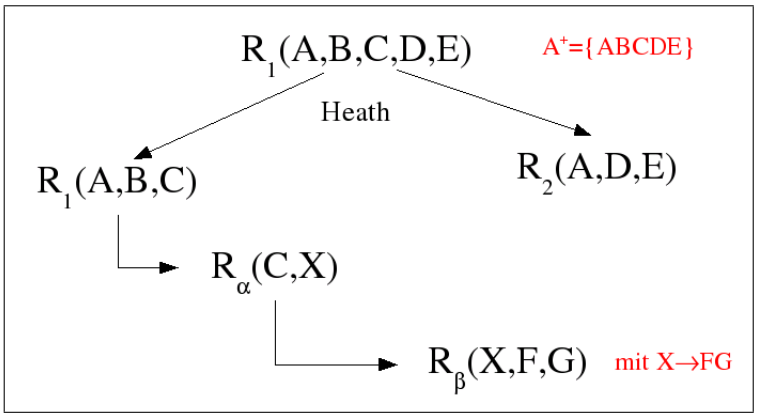

Abbildung 3.12: Dekomposition in BCNF

der einfach-komplexen Beziehung zwischen den Entitäten *ArrayDesign* und *Reporter* bzw. *CompositeSequence* (Kap. 3.2.1, Abb. 3.5) kein Schlüssel hervorgeht, da unterschiedliche Objekte des Entitätstyps *ArrayDesign* gleiche Instanzen des Entitättyps *Reporter* und *CompositeSequence* zugeordnet werden können. Wie bereits hergeleitet ist R(A, B, C, D, E) in BCNF, wie dessen Subrelationen R_1(A, B, C) und R_2(A, D, E). Die Namensgebung der Relationen ist für R_1 *assay_array_design*, für R_2 *bio_material*, für R_α *arrayDesign_compSeq* und für R_β *compSeq_reporter*.

Zu einer Instanz der Entität Experiment Instanzen der Entität ExperimentDesign referenzieren

Aus dem E/R-Modell ist die Kardinalität zwischen den Entitäten *Experiment* und *ExperimentDesign* einfach-komplex (Kap. 3.2.1, Abb. 3.6). Überführt man beide Entitäten in eine Relation R (A, B), mit A als *Experiment* und B als *ExperimentDesign*, so ist keine funktionelle Abhängikeit zu finden, die einen Schlüsselkanditaten hervorruft. Die Begründung liegt darin, dass verschiedene Instanzen des Entitättyps *Experiment* gleiche Objekte der Entität *ExperimentDesign* enthalten können, wie das Tabellenbeispiel in (Abb. 3.13) verdeutlichen soll.

Experiment	*ExperimentDesign*
Alpha	Normal vs. Deseased
Alpha	Treated vs. Untreated
Beta	Normal vs. Deseased
Beta	Treated vs. Untreated

Abbildung 3.13: Tabellenbeispiel zu den Entitäten *Experiment* und *ExperimentDesign*

Obwohl diese Relation nicht einmal in 2. NF ist, da kein Schlüsselkandidat aus ihr hervorgeht, wird sie dennoch in die Datenbank aufgenommen. Die Redundanzanomlie kann hier in Kauf genommen werden, da ein Microarrayexperiment in der Regel aller Fälle nur unter einem Kriterium (Klasse *ExperimentDesign*) durchgeführt wird. Die Tabelle erhält die Bezeichnung *exp_expDesign*.

Zu einer Instanz der Entität BioAssay Instanzen der Klasse ExperimentalFactor, FactorValue, Measurement und Unit abspeichern

Nach dem E/R-Modell besitzt die Entität *ExperimentalFactor* eine Assoziation zur Entität *BioAssay* und *ExperimentDesign* (Kap. 3.2.1, Abb. 3.6). Die Beziehung *ExperimentDesign – ExperimentalFactor* wird in keine Tabelle aufgenommen, da Objekte der Entität *ExperimentalFactor* bereits über die Assoziation *BioAssay – ExperimentalFactor* referenziert werden können. Die Kardinalität gibt aus dem E/R-Modell eine einfach-komplexe Beziehung zwischen den Entitäten *BioAssay* und *ExperimentalFactor* vor (Kap. 3.2.1, Abb. 3.6). Desweiteren ist es denkbar, dass unterschiedliche Objekte des Entitättyps *BioAssay* gleiche Einträge an Objekten der Entität *ExperimentalFactor* zugeordnet sind (Abb. 3.14). Schlussfolgernd kann keine funktionelle Abhängigkeit für diesen Anwendungsfall aufgestellt werden, die einen möglichen Schlüsselkandidaten liefert. Die Relation befindet sich in 1. NF. und wird mit dem Namen *exp_factor* bezeichnet.

BioAssay	*ExperimentalFactor*
Gamma	Timecourse
Gamma	Concentration (Gluose)
Delta	Timecourse

Abbildung 3.14: Tabellenbeispiel zu den Entitäten *BioAssay* und *ExperimentalFactor*

Zu einer Instanz der Entität Experiment die Beschreibung und Publikationen abspeichern

Aus dem E/R-Modell (3.2.1, Abb. 3.7) können für diesen Anwendungsfall folgende funktionelle Abhängikeiten hergeleitet werden:

$$\begin{array}{l|l} Experiment \rightarrow Description & A \rightarrow B \\ BibliographicReference \rightarrow Experiment & C \rightarrow A \end{array}$$

Das Überführen der Attribute A, B und C in eine Relation R (A, B, C) ergibt als Abschlussmenge von A, A^+={A,B}, eine Verletzung der Boyce-Codd Normalform (Abb. 3.15). Mit Anwendung von A^+ auf die Dekomposition von R ergeben sich die Relationen R_1 (A, B) und R_2 (A, C). Beide hergeleiteten Relationen, R_1 und R_2, erfüllen die Kriterien der BCNF. Der Relation R_1 wird die Namensbezeichnung *description* , Relation R_2 die Namensbezeichnung *experiment_bqs* zugeteilt.

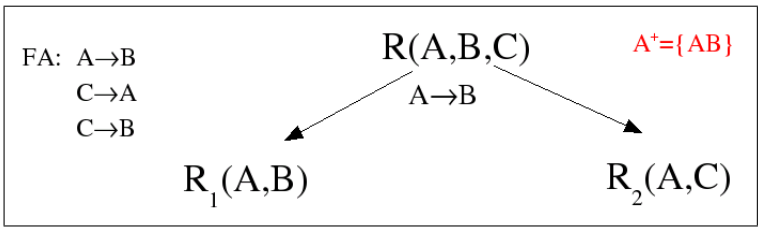

Abbildung 3.15: Dekomposition in BCNF

Zu einer Instanz der Entität BioAssayData die Scanwerte abspeichern

Aus den Kardinalitäten der Entitäten *BioAssay*, *BioAssayData*, *BioAssay*, *CompositeSequence*, *Reporter* und dem Scanwert ergibt sich die Menge an funktionellen Abhängigkeiten:

$BioAssay \rightarrow Experiment$	$A \rightarrow B$
$BioAssayData \rightarrow BioAssay$	$C \rightarrow A$
$BioAssayData \rightarrow Experiment$	$C \rightarrow B$
$CompositeSequence, Reporter, BioAssayData \rightarrow Scanwert$	$DEC \rightarrow F$

Bildet man die Relation R (A, B, C, D, E, F) ergibt A^+={A,B} und ist gleichzeitig eine BCNF-Verletzung (Abb. 3.16). Die Dekomposition von R mit Hilfe der funktionellen Abhängigkeit $A \rightarrow B$ liefert als Ergebnis die Relationen R_1 (A, B), welche bereits in BCNF ist, und R_2 (A, C, D, E, F), die weiter zerlegt werden muss. Mit der funktionellen Abhängigkeit $C \rightarrow A$ ergibt die Abschlussmenge $\{AC\}^+$ eine BCNF-Verletzung. Die Dekomposition von R_2 ergibt die Relationen R_3 und R_4 mit R_3 (C, A) und R_4 (C, D, E, F). Sowohl R_3 und R_4 erfüllen die Kriterien der BCNF. R_1 ist bereits als Tabelle *assay_data_experiment* und R_3 als Tabelle *bioAssayData_bioAssay* bekannt (Kap. 3.7.4 Abb. 3.10). R_4 erhält als Tabellennamen *bio_assay_data*. Die Attribute D und E, welche in ihrer Kombination eindeutig sind, werden zu einem Bezeichner *id* zusammengefasst. Dieser Bezeichner findet sich in der Tabelle *compSeq_reporter* wieder.

3.7.5 Vom E/R-Modell zum Tabellendesign

Das E/R-Modell dient als Vorgabe zum Entwickeln des Tabellendesigns. Die Datenbank erhält den Namen *BioChipDB*.

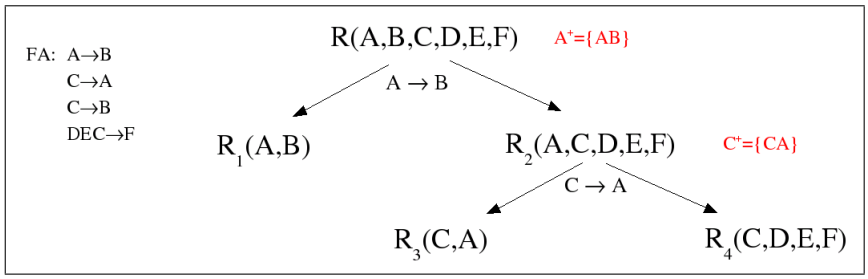

Abbildung 3.16: Dekomposition in BCNF

Tabelle *audit*

Das Anlegen eines Experiments in die Datenbank kann als einmaliger Vorgang interpretiert werden. Ein Experiment muss deshalb in der Datenbank eindeutig identifizierbar sein. Es ist über einen eindeutigen Schlüssel ansprechbar. Das Anlegen eines Experiments wird in der Tabelle *audit* dargestellt (Abb. 3.17). Sie umfasst sechs Attribute, *exp_ID*, *date*, *contact_ID*, *name*, *gsname* und *updated*. Der Schlüssel dieser Tabelle ist das Attribut *exp_ID*, welcher ein positiver, autoinkrementeller Integerwert ist, der ausschließlich seitens der Datenbank inkrementiert werden kann. Im Attribut *date* wird das Datum abgelegt, an dem ein Datensatz für ein Microarrayexperiment abgespeichert wurde. Für jedes Objekt der Entität *Experiment* wird in der Datenbank vom Benutzer ein Name vergeben und im Attribut *name* festgehalten, dem in der Tabelle keine Eindeutigkeit zugeordnet werden kann. Das Attribut *gsname* ist ein von einer weiteren Inhousedatenbank automatisch vergebener Name. Für die Protokollierung ist es ebenfalls nötig zu wissen, welche Person den Datensatz abgespeichert hat. Diese Information wird im Attribut *contact_ID* gespeichert, welches ein Integerwert ist. Wird ein Datensatz nachträglich bearbeitet, ist das Attribut *updated* angelegt worden, in dem das Datum der Veränderung niedergeschrieben wird.

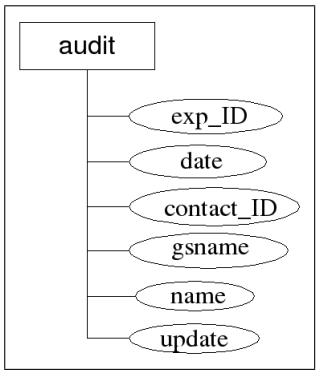

Abbildung 3.17: Tabelle *audit*

Tabelle *contact*

Jeder Datensatz, der in die Datenbank eingetragen wird, muss einer Person zugewiesen werden können. Da der Name einer Person bzw. der eines Instituts nicht immer eindeutig ist, ist dafür eine separate Tabelle anzulegen. In der Tabelle *contact* (Abb. 3.18) werden die Attribute *contact_ID*, *address*, *phone*, *email*, *fax*, *lastname*, *firstname* und *midinitials* aufgenommen. Im Attribute *address* wird die Anschrift, im Attribut *email* die Emailadresse, im Attribut *fax* die Faxnummer, im Attribut *lastname* der Nachname, im Attribut *firstname* der Vorname einer Person und im Attribut *midinitials* die Initialien von Vor- und Nachnamen festgehalten.

Tabelle *assay_data_experiment*

In der Tabelle *assay_data_experiment* (Abb. 3.19) sind zwei Attribute angelegt: *bio_assay_ID* und *exp_ID*. Nach dem Klassendiagramm (Kap. 3.1, Abb. 3.1) besteht die Klasse *Experiment* aus Objekten des Typs *BioAssay*. Aus dem Resultat des E/R-Modells ist die Beziehung zwischen den Entitäten *Experiment* und *BioAssay* einfach-komplex. Daraus ergibt sich für das Design der Tabelle *assay_data_experiment*, dass pro Entität *BioAssay* ein eigener Schlüssel vergeben werden muss, der im Attribut *bio_assay_ID*

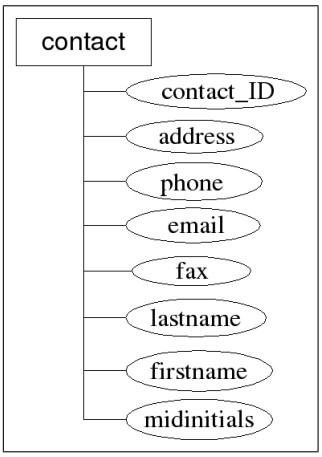

Abbildung 3.18: Tabelle *contact*

abgelegt wird, wie es aus der BCNF-Zerlegung hervorgeht (Kap. 3.7.4 Abb. 3.10).

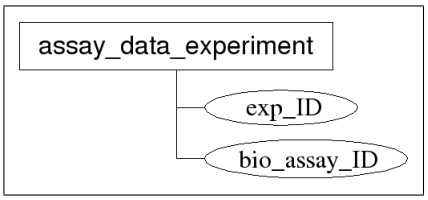

Abbildung 3.19: Tabelle *assay_data_experiment*

Tabelle *compSeq_reporter*

Für das Abspeichern der Entitäten *CompositeSequence* und *Reporter* wurde gemäß der BCNF-Zerlegung (Kap. 3.7.4, Abb. 3.12) die Tabelle *compSeq_reporter* angelegt (Abb. 3.20). Die biologische Beziehung zwischen einem Gen und seinem Protein ist unter anderem abhängig von der Splicevariante des Gens. In Bezug auf das Modell bedeutet das, dass es zu einer Instanz des Typs *CompositeSequence* mehrere Instanzen des Typs *Reporter* gibt, jedoch zu einer Instanz *Reporter* genau eine Instanz *CompositeSequence*.

Jeder Eintrag in dieser Tabelle ist über den Primärschlüssel *id* referenzierbar. Mit dieser Tabelle ist beabsichtigt, alle Gene in einer Tabelle zentral abzuspeichern um redundante Einträge zu vermeiden.

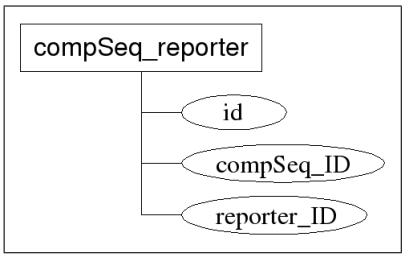

Abbildung 3.20: Tabelle *compSeq_reporter*

Tabelle *bio_assay_data*

Jegliche Scanwerte, die von einem Gen pro Spot auf dem Microarry geliefert werden, werden in der Tabelle *bio_assay_data* abgespeichert (Abb. 3.21). Für die Scanwerte ist das Attribut *value* angelegt worden. Ist ein Scanwert für ein Gen nicht vorhanden, wird der Eintrag auf *N.N* gesetzt, was für *not a number* steht. Damit ein Scanwert einem Gen und der zugehörigen Entität *BioAssayData* zugeordnet werden kann, sind die Attribute *bio_assay_data_ID* und *id* aus den Tabellen *compSeq_reporter* und *bioAssayData_bioAssay* notwendig. Das Attribut *bio_assay_data_ID* ist ein eindeutiger Schlüssel für ein Objekt des Entitätstyps *BioAssayData*, und das Attribut *id* für die Entitätenkombination *CompositeSequence* und *Reporter*.

Tabelle *bioAssayData_format*

Die Scanwerte zu einem Microarrayexperiment können in verschiedenen Formaten vorliegen. Dazu wurde in der Tabelle *bioAssayData_format* (Abb. 3.22) eine Spalte namens *format* angelegt, die jene Information niederschreibt. Die Zuordnung zu einem Objekt der Entität *BioAssayData* erfolgt über den

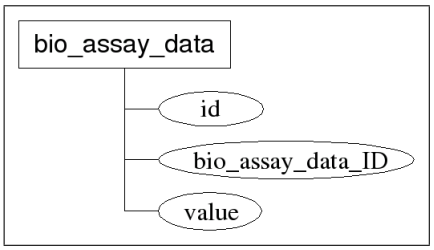

Abbildung 3.21: Tabelle *bio_assay_data*

Schlüssel *bio_assay_data_ID*.

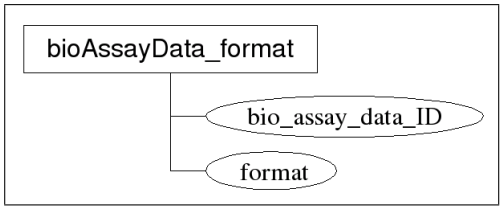

Abbildung 3.22: Tabelle *bioAssayData_format*

Tabelle *array_identifier*

Für das Abspeichern eines Microarraytyps ist die Tabelle *array_identifier* (Abb. 3.23) angelegt worden, in der in der gesamten Datenbank die Microarrayhersteller abgespeichert werden. Das Attribut *array_ID* ist Schlüssel dieser Tabelle und im Attribut *ontology* wird der Name des Arrayherstellers niedergeschrieben.

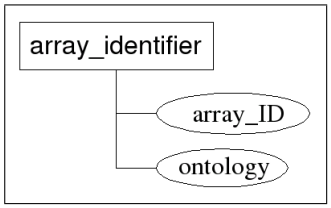

Abbildung 3.23: Tabelle *array_identifier*

Tabelle *arrayDesign*

Da die Kardinalität zwischen den Entitäten *ArrayDesign* und *Array* einfach-einfach ist (Kap. 3.2 Abb. 3.5), ist sie in der Tabelle *arrayDesign* festzuhalten (Abb. 3.24). Die Tabelle besteht aus den Attributen *arrayDesign_ID*, *array_ID*, *version*, *number_of_features* und *prop_ID*. Jeder neue Eintrag für ein Objekt des Entitättyps *ArrayDesign* wird über den Schlüssel *arrayDesign_ID* referenziert. Die Zuordnung zum physikalischen Array, und damit zur Entität *Array* erfolgt über den Schlüssel *array_ID*. Im Attribut *version* wird die Version des Arrays und im Attribut *number_of_features* die Spotanzahl des Genchips abgespeichert. Sollte eine firmeninterne Bezeichnung des Genchips vorhanden sein, so wird diese im Attribut *prop_ID* festgehalten.

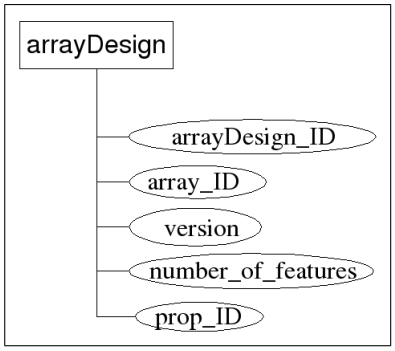

Abbildung 3.24: Tabelle *arrayDesign*

Tabelle *assay_array_design*

In der Tabelle *assay_array_design* (Abb. 3.25) werden die Beziehungsketten zwischen den Entitäten *BioAssay – Array* und *Array – ArrayDesign* zusammengeführt, wie es aus der BCNF-Zerlegung (Kap. 3.7.4, Abb. 3.12) vorgesehen ist. Pro gespeichertem Objekt der Entität *BioAssay* werden die Schlüssel für die Objekte der Entitäten *Array* und *ArrayDesign* aus den Tabellen *array_identifier* und *arrayDesign* entnommen und aufgelistet. Somit

werden in der Tabelle *assay_array_design* nur Schlüsselreferenzen niedergeschrieben.

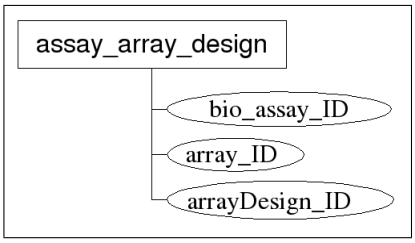

Abbildung 3.25: Tabelle *assay_array_design*

Tabelle *description*

Aufgabe der Tabelle *descripiton* ist es, die zu einem Objekt der Entität *Experiment* mitgeführte Beschreibung aus dem Attribut *explanation* der Klasse *Description* festzuhalten. Da die Komplexität zwischen den Entitäten *Experiment* und *Description* einfach-einfach ist (Kap. 3.2 Abb. 3.2), gestaltet sich das Abspeichern in dieser Tabelle sehr einfach. Die Entität *Description*, die in der Tabelle *description* abgelegt wird, verfügt über die Spalten *exp_ID* und *explanation*. Schlüssel dieser Tabelle ist das Attribut *exp_ID*, wie die BCNF-Zerlegung (Kap. 3.7.4, Abb. 3.15) vorgibt.

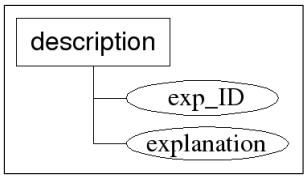

Abbildung 3.26: Tabelle *description*

Tabelle *experiment_bqs*

In der Tabelle *experiment_bqs* (Abb. 3.27) werden die Schlüsselreferenzen der Entitäten *Experiment* und *BibliograhpicReference* zusammengeführt. Das

Attribut *exp_ID* verweist auf den vergebenen Identifikationsschlüssel für ein abgespeichertes Microarrayexperiment, und das Attribut *bqs_ID* indiziert die zu einem Objekt der Klasse *Experiment* mitgeführten Instanzen der Klasse *BibliographicReference*.

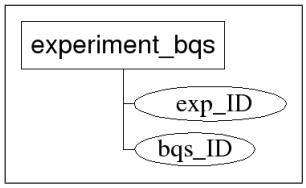

Abbildung 3.27: Tabelle *experiment_bqs*

Tabelle *bqs*

Ist nun ein Dateneintrag für eine Publikation in der Tabelle *experiment_bqs* erfolgreich abgelegt worden, ist der vergebene Schlüssel in die Tabelle *bqs* (Abb. 3.28) einzutragen. Der Titel einer Publikation wird in der Spalte *title*, der Autor in *authors*, der Abstrakt in *abstract*, die PubMedID in *pubMedID* und die Referenz in *reference* abgespeichert.

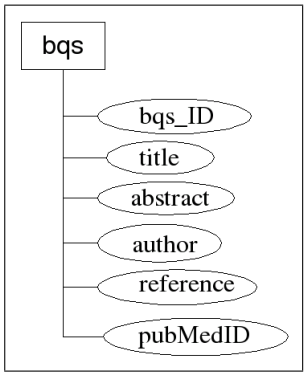

Abbildung 3.28: Tabelle *bqs*

Tabelle *bio_source*

Mit der Tabelle *bio_source* (Abb. 3.29) wurde in der Datenbank eine zentrale Stelle zum Abspeichern an biologischem Probenmaterial geschaffen, die jeweils pro Microarraydatensatz anfallen. Der Aufbau der Tabelle besteht aus den Attributen *bio_source_ID* und *ontology*. Dabei fungiert das Attribut *bio_source_ID* als Schlüssel, das Attribut *ontology* definiert das für die Entität *BioSource* definierte Fachvokabular.

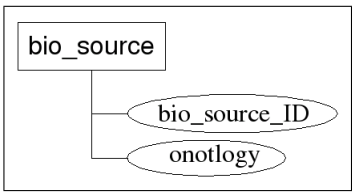

Abbildung 3.29: Tabelle *bio_source*

Tabelle *bio_material*

Im Klassenmodell ist eine Instanz der Klasse *BioMaterial* eine Aggregation aus den Objekten der Klasse *BioSource* und *MaterialType*. In der Tabelle *bio_material* (Abb. 3.30) werden die Beziehungen zwischen den Entitäten *BioMaterial*, *BioSource* und *MaterialType* zusammengeführt. Die Tabelle *bio_material* besteht deshalb ausschließlich aus Schlüsselreferenzen der genannten Entitäten.

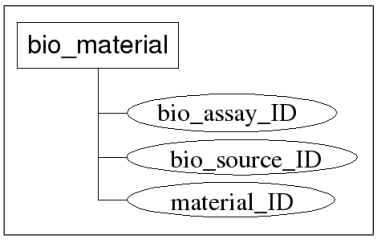

Abbildung 3.30: Tabelle *bio_material*

Tabelle *exp_expDesign*

In der Tabelle *exp_expDesign* werden nach dem Klassenmodell der OMG (Kap. 3.1, Abb. 3.1) die Objekte der Klasse *ExperimentDesign* abgespeichert. Die Beschreibung, zu einer Instanz dieser Klasse *ExperimentDesign* wird in der Spalte *exp_design_ontology* festgehalten.

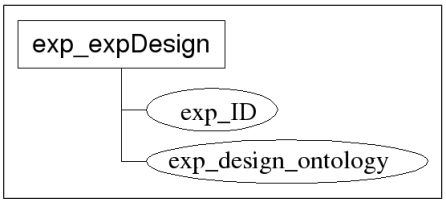

Abbildung 3.31: Tabelle *exp_expDesign*

Tabelle *exp_factor*

Das Abspeichern einer Instanz der Klasse *ExperimentalFactor* ist in der Tabelle *exp_factor* (Abb. 3.32) vorgesehen. Desweiteren werden in dieser Tabelle die Klassen *Measurement* und *Unit* festgehalten. Ein Objekt der Klasse *Measurement* wird in den Attributen *kind* und *type*, ein Objekt der Klasse *Unit* im Attribut *unit* abgespeichert. Die mitgeführten Ontologien zu den Entitäten *ExperimentDesign* und *ExperimentalFactor* werden in den Attributen *expDesign* und *expFactor* abgelegt. Schlüssel dieser Tabelle ist das Attribut *bio_assay_ID*.

Tabelle *material_type*

In der Tabelle *material_type* (Abb. 3.33) wird das Eigenschaftsfeld *ontology* aus der Klasse *MaterialType* abgespeichert und mit Hilfe des Schlüssels *material_ID* indiziert.

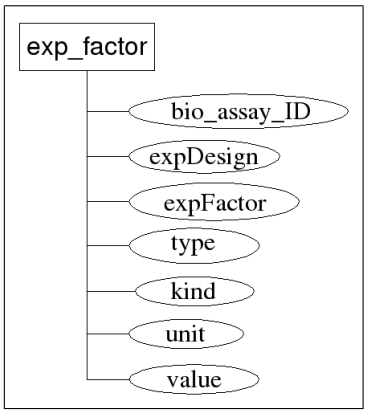

Abbildung 3.32: Tabelle *exp_factor*

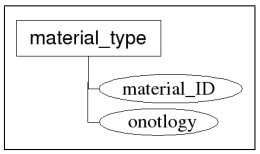

Abbildung 3.33: Tabelle *material_type*

Tabelle *arrayDesign_compSeq*

Da keine Instanz der Klasse *ArrayDesign* in der Datenbank redundant abgespeichert werden soll, wird mit der Tabelle *arrayDesign_compSeq* (Abb. 3.34) eine zentrale Stelle geschaffen, die das Design eines physikalischen Arrays speichert. Dazu ist das Attribut *arrayDesign_ID* aufzunehmen, welches zu einem physikalischen Array die zugehörige Instanz der Klasse *ArrayDesign* zu einem Objekt der Klasse *BioAssay* abspeichert. Die Gene zu einem Array werden mit Hilfe des Schlüssels *id* aus der Tabelle *compSeq_reporter* im Attribut *compSeq_ID* niedergeschrieben.

Tabelle *bioAssayData_bioAssay*

Da jedem Objekt der Klasse *BioAssay* mehrere Instanzen der Klasse *BioAssayData* zugeordnet werden können, ist die einfach-komplexe Beziehung

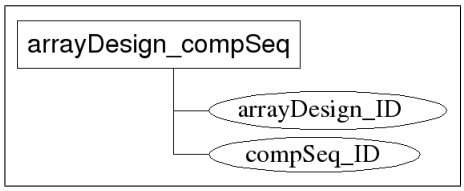

Abbildung 3.34: Tabelle *arrayDesign_compSeq*

zwischen diesen Klassen in der Tabelle *bioAssayData_bioAssay* (Abb. 3.35) abzulegen. Das Attribut *bio_assay_data_ID* ist Schlüssel, wie aus der BCNF-Zerlegung (Kap. 3.7.4, Abb. 3.10) hervorgeht, das Attribut *bio_assay_ID* indiziert die zu einem Objekt der Entität *BioAssay* zugehörigen Instanzen der Entität *BioAssayData*.

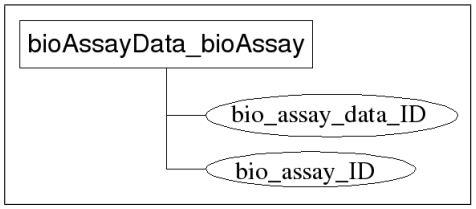

Abbildung 3.35: Tabelle *bioAssayData_bioAssay*

3.8 Tabellenerzeugung in MySQL

Im folgenden Abschnitt wird die Tabellenimplementierung in Abhängigkeit von MySQL beschrieben. Pro Tabelle, die in der Datenbank *BioChipDB* aufgenommen wurde, wird auf Primärschlüssel, Fremdschlüssel, Indexreferenzierung und Datentypen der Spalten sowie Referenzierung auf andere Tabellen Stellung genommen. Die Tabellengestaltung wird durch die Faktoren Speicherplatz, Speicherzeit und Suchgeschwindigkeit determiniert. Neben den Tabellenattributen, die Festplattenkapazität in Anspruch nehmen, wird durch die Vergabe an Primär- und Indexschlüsseln zusätzlich Speichervolumen in Anspruch genommen. Aufgrund der großen Speicherkapazität

heutiger Festplatten (im 3-stelligen Gigabytebereich) kann dieser Aspekt als unproblematisch betrachtet werden. Die Suchzeit ist in Zusammenhang mit vergebenen Primär- und Indexschlüsseln in einer Tabelle zu sehen. Das Suchen über eine verschlüsselte Spalte ist um den Faktor 1000 schneller als über unverschlüsselte Spalten [8], da für jedes Tupel in der Tabelle ein Ergebnisvergleich durchgeführt werden muss. Die Speicherzeit ist um so länger, je mehr Primär- bzw. Indexschlüssel in einer Tabelle vergeben werden, da neben den Datenwerten dafür zusätzlich Speicherplatz angelegt werden muss.

Tabelle *audit*

Die Eindeutigkeit eines Microarrayexperiments ist durch den Primärschlüssel *exp_ID* bestimmt. Die Attribute *date* und *updated*, welche Datumsangaben darstellen, sind dem Datentyp *DATETIME* zugeordnet. Die Initialisierung dieser Attribute wird clientseitig über den Methodenaufruf *System.currentTime()* ausgeführt. Das Attribut *contact_ID* ist definiert als nichtnegativer Integerwert, da nach dem Modell ein Microarrayexperiment durch eine Instanz des Objekttyps *Contact* eingetragen wird. Das bedeutet, dass es durch die Implementierungsstruktur *not null* erweitert werden muss. Der Name einer Instanz der Entität *Experiment* ist eine Zeichenkette mit maximaler Länge 20. Da die Suche nach einem Experiment über das Eigenschaftsfeld *name* für den Benutzer intuitiver, von Seiten der Datenbank jedoch nicht eindeutig ist, wurde diese Spalte als Index der Tabelle *audit* definiert.

Tabelle *contact*

Da in der Tabelle *contact* lediglich Informationen bezüglich einer Person oder eines Instituts abgespeichert werden, verfügt diese Tabelle nur über einen Primärschlüssel (*contact_ID*) und keine weiteren Indexspalten. Alle restlichen Spalten entsprechen dem SQL-Datentyp *varchar* unterschiedli-

cher Länge. Da in der Tabelle *audit* der Primärschlüssel der Tabelle *contact* abgespeichert wird, besitzt die Tabelle *audit* eine Referenz auf die Tabelle *contact*. Die Tabelle *contact* ist jedoch unabhängig von der Tabelle *audit* und verfügt in der Tabellenimplementierung über keinerlei Referenzierung bzw. Kaskadierung. Die Konsequenz daraus ist, dass die Tabelle *contact* in der Datenbank eine unabhängige Stellung einnimmt und im Falle des Löschens eines Dateneintrags in der Tabelle *audit* davon unberührt bleibt.

Tabelle *assay_data_experiment*

Der Primärschlüssel der Tabelle *audit* fungiert in dieser Tabelle als Fremdschlüssel. Gleichzeitig wird die Spalte, in der der Fremdschlüssel abgespeichert wird, als Index definiert. Die Begründung dafür liegt in der Suchgeschwindigkeit, da über das Attribut *exp_ID* die Scanwerte zu einer Instanz der Entität *BioAssayData* aufgelistet werden sollen. Mit dem Fremdschlüssel referenziert die Tabelle *assay_data_experiment* auf die Tabelle *audit* und ist zugleich von ihr abhängig. Wird ein Dateneintrag zu einem Microarrayexperiment in der Tabelle *audit* gelöscht, so müssen auch alle zugehörigen Dateneinträge aus der Tabelle *assay_data_experiment* entnommen werden. Dazu ist die Kaskadierung dieser Tabelle hinzuzufügen.

Tabelle *bioAssayData_bioAssay*

Die einfach-komplexe Beziehung zwischen den Entitäten *BioAssay* und *BioAssayData* definiert das Attribut *bio_assay_data_ID* als Primärschlüssel, und das Attribut *bio_assay_ID* als Fremdschlüssel. Der Femdschlüssel zeigt somit auf die Tabelle *assay_data_experiment*. Wird nun ein Eintrag aus der Tabelle *assay_data_experiment* gelöscht, so sind davon auch die zugehörigen Einträge in der Tabelle *bioAssayData_bioAssay* betroffen. Diese Vekettung der Tabellen wird durch die Kaskadierung implementiert.

Tabelle *bioAssayData_format*

Das Attribut *bio_assay_data_ID*, welches der Primärschlüssel der Tabelle *bioAssayData_bioAssay* ist, wird in der Tabelle *bioAssayData_format* als Fremdschlüssel aufgenommen. Die dadurch entstandene Referenz auf die Tabelle *bioAssayData_bioAssay* führt dazu, dass das Entfernen eines Dateneintrags in dieser Tabelle ebenfalls die Tabelle *bioAssayData_format* betrifft. Das Attribut *format* ist eine Zeichenkette der Länge 20 und erhält gleichzeitig einen Index, da dies ein mögliches Suchkriterium darstellt. Das Entfernen eines Dateneintrags aus der Tabelle *bioAssayData_bioAssay* impliziert das Löschen der zugehörigen Tupel in der Tabelle *bioAssayData_format*.

Tabelle *bio_assay_data*

Die Tabelle *bio_assay_data* hat mit den Attributen *id* und *bio_assay_data_ID* Referenzen auf die Primärschlüssel der Tabellen *compSeq_reporter* und *bioAssayData_bioAssay*. Beide Attribute werden in dieser Tabelle als Fremdschlüssel angelegt. Lediglich der Fremdschlüssel *bio_assay_data_ID* referenziert und kaskadiert auf die Tabelle *bioAssayData_bioAssay*, da mit dem Löschen eines Datentupels in der Tabelle *bioAssayData_bioAssay* auch die dazu referenzierten Scanwerte betroffen sind. Der Fremdschlüssel verfügt über keinerlei Kaskadierung, da in der Tabelle *compSeq_reporter* keine Tupel entfernt werden dürfen. Der Grund dafür ist, dass wenn Datentupel aus der Tabelle *compSeq_reporter* gelöscht werden, die Scanwerte zu einem Gen nicht mehr zugeordnet werden können. Wenn jedoch Datentupel aus der Tabelle *bioAssayData_bioAssay* entfernt werden, betrifft dies auch die Tabelle *bio_assay_data*, da nur dann Scanwerte existieren, wenn eine zugehörige *bio_assay_data_ID* vergeben wurde.

Tabelle *compSeq_reporter*

Die zentrale Rolle der Tabelle *compSeq_reporter* macht sie unabängig von allen anderen Tabellen in der Datenbank. Da in dieser Tabelle keine Redundanz an Geneinträgen vorhanden sein darf, werden seitens der Datenbank in der Tabellenimplementierung die Attribute *compSeq_ID* und *reporter_ID* durch das Schlüsselwort *unique* eine Eindeutigkeit zugeordnet. Die Attribute selbst sind vom Datentyp *varchar* und werden indiziert. Primärschlüssel der Tabelle ist das Attribut *id*, ein autoinkrementelles Integer.

Tabelle *array_identifier*

Primärschlüssel dieser Tabelle ist das Attribut *array_ID*, welches dem Datentyp Integer zugeordnet wird. Da in dem Attribut *identifier* Arraytypen abgespeichert werden, wird dieses Attribut als Zeichenkette der Länge 30 festgelegt. Da der Arraytyp in der Datenbank ein mögliches Suchkriterium ist, wird die Spalte *identifier* einem Index übergeben.

Tabelle *arrayDesign*

Für die Implementierung der Tabelle *arrayDesign* ist mit dem Fremdschlüssel *array_ID* eine Referenz auf die Tabelle *array_identifier* angelegt worden. Desweiteren wird durch Kaskadierung eine Zusicherung getroffen, dass wenn ein Tupel aus der Tabelle *array_identifier* entfernt wird, dies anschließend auch in der Tabelle *arrayDesign* ausgeführt wird. Da ein Suchkriterium in der Datenbank die Version eines Genchips sein kann, wird die Spalte *version* als Schlüssel indiziert. Das Attribut *number_of_features* ist vom Typ Integer und darf keinen *null*-Wert besitzen, da jedem Array nach dem Klassenmodell eine Spotanzahl zugeordnet ist. Das Attribut *prop_ID* ist vom Typ *varchar* mit Länge 20.

Tabelle *assay_array_design*

Die Tabelle *assay_array_design* verfügt über die Fremdschlüssel *bio_assay_ID*, *array_ID* und *array_design_ID*, welche in den Tabellen *assay_data_experiment*, *arrray_identifier* und *arrayDesign* als Primärschlüssel definiert sind. Zur Optimierung der Suchgeschwindigkeit werden alle Fremdschlüssel indiziert. Von den Fremdschlüsseln besitzt lediglich *bio_assay_ID* eine Referenz auf die Tabelle *assay_data_experiment* und verfügt gleichzeitig über die Kaskadieroperationen *on delete* und *on update*. Wird ein Dateneintrag aus der Tabelle *assay_data_experiment* für eine Entität *BioAssay* gelöscht, so betrifft dies auch die daran angeknüpften Informationen bzgl. der Entitäten *ArrayDesign* und *Array*.

Tabelle *bio_source*

In der Tabelle *bio_source* ist das Attribut *bio_source_ID* als Primärschlüssel und das Attribut *ontology* als Zeichenkette der Länge 20 gesetzt. Die Tabelle referenziert auf keine anderen Tabellen, da in ihr Fachvokabular abgespeichert ist. Dennoch ist es wünschenswert, keine redundanten Ontologieeinträge zu einem gegebenen Primärschlüssel in dieser Tabelle abzuspeichern. Aus diesem Grund wird das Attribut *ontology* dem Schlüssel *unique* übergeben.

Tabelle *material_type*

Die Implementierung der Tabelle *material_type* ist analog zur Tabelle *bio_source*. Für das Attribut *ontology* sind keine Mehrfacheinträge zugelassen. Es gilt ebenfalls die Ontologieeinträge dem Schlüssel *unique* zu übergeben.

Tabelle *bio_material*

Die Tabelle *bio_material* ist, wie die Tabelle *assay_data_experiment* aus mehreren Primärschlüsseln anderer Tabellen zusammengesetzt, die gleichzeitig

Fremdschlüssel sind. Das Attribut *bio_assay_ID* referenziert auf die Tabelle *assay_data_experiment*, *material_ID* auf *material_type* und *bio_source_ID* auf *bio_source*. Da in den Tabellen *material_type* und *bio_source* Fachvokabular abgespeichert ist, welches einmal in die Tabelle abgelegt nicht mehr entnommen werden darf, ist es nicht notwendig, diese beiden Fremdschlüssel mit *update*- oder *delete*-Operationen zu versehen. Jedoch der Fremdschlüssel *bio_assay_ID* muss diese Operation aufnehmen, da beim Löschen eines Tupels aus der Tabelle *assay_data_experiment* die zugehörigen Einträge in der Tabelle *bio_material* gelöscht werden.

Tabelle *arrayDesign_compSeq*

Da in der Tabelle *arrayDesign_compSeq* Informationen aus den Tabellen *arrayDesign* und *compSeq_reporter* zusammengefasst werden, besteht diese Tabelle ausschließlich aus dessen Primärschlüsseln und referenziert gleichzeitig auf dessen Tabellen. In Bezug auf das Klassenmodell speichert diese Tabelle zu einer Instanz der Klasse *ArrayDesign*, die aus einem Array an Objekten der Klassen *CompositeSequence* und *Reporter* besteht, die zu einem Tupel (*CompositeSequence*, *Reporter*) zugehörigen Identifikationsschlüssel aus der Tabelle *compSeq_reporter* ab.

Abschließend ist mit Abb. 3.36 eine Zusammenfassung der Tabellenreferenzierung der Datenbank *BioChipDB* gegeben.

3.8.1 Softwareseite: Implementierung des Klassenmodells

Nachdem das Datenbankdesign abgeschlossen ist, erfolgt das Einspielen von Microarraydatensätzen in die Datenbank, welche über das Internet bezogen werden. Da die im Internet zugänglichen Micorarraydatenbanken kein einheitliches Schema besitzen, sind die Daten dem Datenbankschema anzupassen. Die Implementierung erfolgte nach folgendem Flussschema:

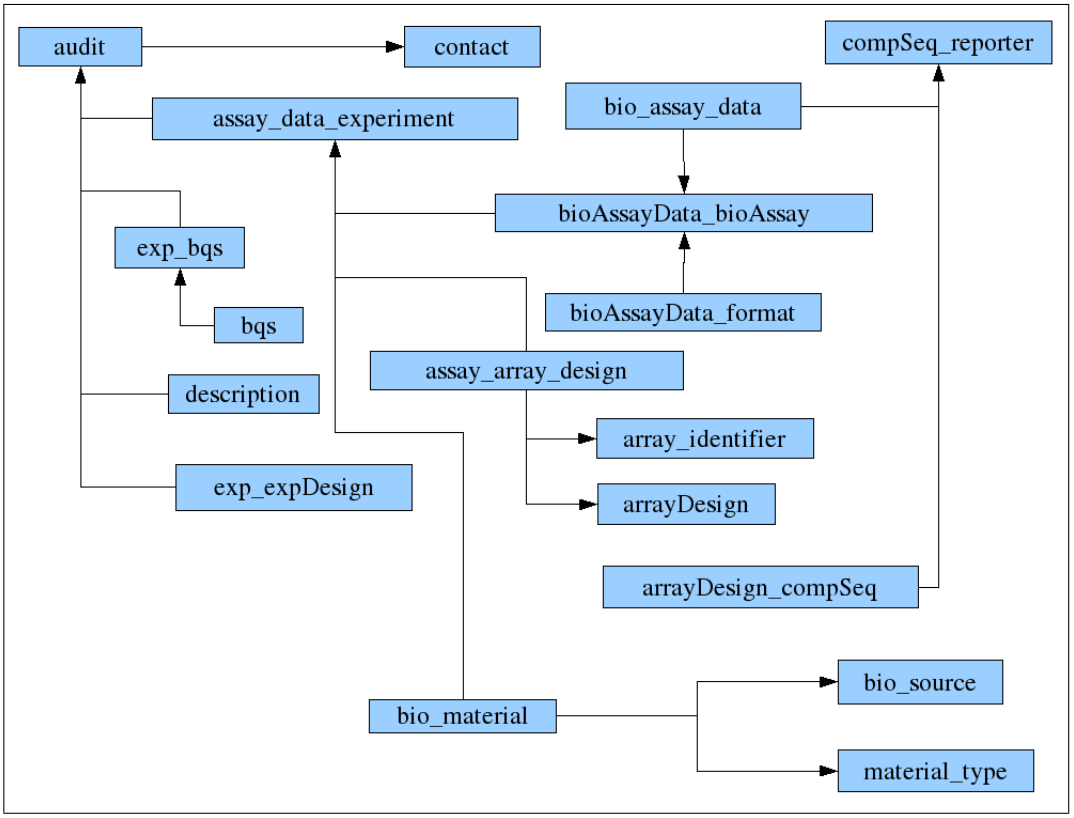

Abbildung 3.36: Tabellenreferenzierungen in *BioChipDB*

1. Implementierung des Klassenmodells
2. Implementierung eines Parsers, der aus Dateien öffentlicher Datenbanken ein Objekt der Klasse *Experiment* zusammenstellt
3. Implementierung eines Programms zum Abspeichern einer Instanz der Klasse *Experiment*

3.8.2 Implementierung des Klassenmodells

Das Klassenmodell, welches aus dem Genexpressionsspezifikation der OMG [5] entwickelt wurde, kann direkt in die objektorientierte Programmiersprache Java überführt werden. Klassennamen, Attribute und Abhängigkeiten

entsprechen dem Klassendiagramm.

3.8.3 Implementierung der Klassen *Filter*, *Parser*, *Composer*

Die Komplexität zur Bildung eines Objekts der Klasse *Experiment* wird in einer Schrittkette auf die Klassen *Filter*, *Parser* und *Composer* aufgeteilt.

Klasse *Filter*

Die Aufgabe der Klasse Filter (Abb. 3.37) ist es, die zu einem Microarrayexperiment abgespeicherten Daten zu filtern. Es sollen hier jene Informationen aus den Dateien extrahiert werden, die für ein Objekt der Klasse *Experiment* relevant sind. Aufgrund verschiedener Datenbanken ist es nicht möglich, die Klasse *Filter* zu generalisieren. Es sind vielmehr spezifische Klassen von *Filter* zu implementieren. Die programmatische Lösung liegt darin, die Klasse *Filter* als abstrakt zu definieren. Pro öffentlicher Datenbank gibt es eine eigenst implementierte Klasse von *Filter*.

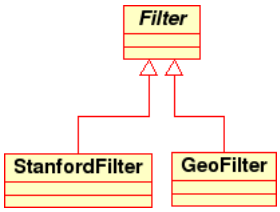

Abbildung 3.37: Klassendiagramm für *Filter*, *StanfordFilter* und *GeoFilter*

Klasse *Parser*

Aufgabe der Klasse *Parser* (Abb. 3.38) ist es, die aus der Klasse *Filter* grobstrukturierten Daten nach klassenabhängigen Informationen zu untersuchen. Da die Implementierung der Klasse *Parser* von der Datenstruktur und damit von der öffentlichen Datenbank abhängt, ist ein Parser datenbankspezifisch

zu implementieren. In Bezug auf die Softwarearchitektur bedeutet das, dass die Klasse *Parser* abstrakt gehalten wird und pro Datenbank eine Subklasse der Klasse *Parser* existiert.

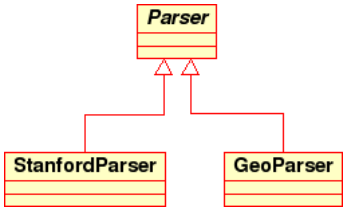

Abbildung 3.38: Klassendiagramm für *Parser*, *StanfordParser* und *GeoParser*

Klasse *Composer*

Die Klasse *Composer* (Abb. 3.39) hat die Aufgabe eine Instanz der Klasse *Experiment* aus den Ergebnissen eines Parsers zu erzeugen. Das instanzierte Objekt der Klasse *Experiment* wird einem Objekt der Klasse *MySQLExperimentStorer* übergeben, das das Experimentobjekt in die Datenbank einspielt. Die Klasse *Composer* ist, wie die Klassen *Parser* und *Filter*, eine abstrakte Klasse, welche spezialisiert werden muss. Pro Internetdatenbank wird eine Subklasse implementiert.

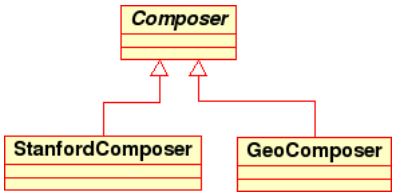

Abbildung 3.39: Klassendiagramm *Composer*, *StanfordComposer* und *GeoComposer*

3.8.4 Implementierung des Speicherprogramms

Nachdem eine Instanz der Klasse *Experiment* durch eine der Subklassen von *Composer* erzeugt wurde, kann das Objekt an die Methode *store(Experiment experiment)* der Klasse *MySQLExperimentAdaptor* übergeben werden, einer Klasse die die Fähigkeit besitzt, das Experimentobjekt der Datenbank *BioChipDB* zuzuordnen. Das Argument dieser Methode wird an die Klasse *MySQLExperimentStorer* weitergegben, die die in Abb. 3.40 angegebenen Speichermethoden besitzt.

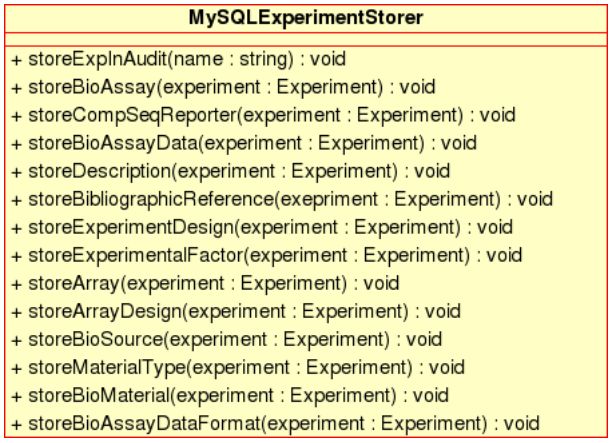

Abbildung 3.40: Klassendiagramm für *MySQLExperimentStorer*

3.8.5 *ExpLoader*: Die Applikation zum Abspeichern eines Microarrayexperiments

Zum Abspeichern eines Microarraydatensatzes, welcher aus dem Internet bezogen wurde, sind alle notwendigen Java-Klassendateien in einer Jar-Datei namens *ExpLoader.jar* zusammengefasst worden. Die Applikation wird konsolenbasiert ausgeführt. Ihr sind insgesamt 6 Parameter zu übergeben. Der erste Parameter ist der Zeichenkette *–store* gleichzusetzen. Als zweiter Parameter ist der Datenbankname der Microarraydatenbank zu übergeben, aus dem der Datensatz bezogen wurde. Der dritte Parameter gibt den Orga-

nismus an. Mit dem vierten Parameter ist das Probenmaterial anzugeben, welches auf den Chip gegeben wurde. Der fünfte Parameter gibt den durch den Benutzer vergebenen Experimentnamen an. Der sechste und letzte Parameter gibt den Dateinamen des Microarraydatensatzes an. Im Falle eines Stanford-Datensatzes ist nur der absolute Verzeichnispfad anzugeben, unter der Bedingung, dass alle Stanforddateien in einem Ordner abgespeichert wurden. Für einen GEO-Datensatz ist der absolute Pfad der Datei anzugeben.

3.9 Microarraydatenbanken im Internet

In die Datenbank *BioChipDB* wurden Datensätze aus den Internetdatenbanken *Stanford Microarray Database* [16] und *Gene Expression Omnibus* [2] abgespeichert. Im folgenden Abschnitt soll anhand eines Datensatzes aus der Datenbank von Stanford die Interpretierbarkeit der Daten in Bezug auf das Klassenmodell (Kap. 3.1, Abb. 3.1) verdeutlicht werden.

3.9.1 SMD: Stanford Microarray Database

Genexpressionsdaten, die aus der Datenbank von Stanford bezogen werden können, sind in drei Dateien aufgeteilt:

- *Publicationfile*
- *Metafile*
- *Datafile*

Publicationfile

Der Inhalt dieser Datei (Abb. 3.41) ist in XML-artiger Struktur aufgebaut. An erster Stelle in der Hierarchie steht der Knoten *publication*. Die Zeile

beginnend mit *Citation* stellt im Modell ein Objekt der Klasse *BibliographicReference* dar. Die darauffolgende Zeile, die als *Title* bezeichnet wird, ist in einem Objekt der Klasse *BibliographicReference* dem Attribut *title* gleichzusetzen.

```
<publication>
!Citation=Gerber et. al. (2004) PLOS Biology, 2(3),342-354
!Title=Extensive Association of Functionally and Cytotopically
        Related mRNAs with Puf Family RNA-Binding Proteins in Yeast
<experiment_set>
           !Name=Puf3delta vs. WT, SC_glycerol, Gerber AP.
           !ExptSetNo=2623
           !Description=Three independent experiments: S. cervisiae
                        wild-type (BY4741) and Puf3 mutant cells were
                        grown in minimal medium supplemented with 3%
                        glycerol. Cells were harvested in mid-log
                        phase by centrifugation and total RNA was
                        prepared by hot-phenol extraction. cDNA was
                        prepared with oligo-dT and using a mixture of
                        amino-allyl dUTP and dNTPs, fluorescently
                        labeled (Cy3= wild-type, Cy5 = mutant) and
                        hybridize on S. cerevisiae DNA microarray.
     </experiment_set>
</publication>
```

Abbildung 3.41: Dateiinhalt Publicationfile

Metafile

Das Metafile (Abb. 3.42) ist ebenfalls in eine XML-artige Struktur gepackt. In dieser Datei werden die zu einem Experiment enthaltenen Instanzen der Klasse *BioAssay* beschrieben. Im Modell entspricht der Knoten *experiment_set* der Klasse *Experiment*, und die Kindknoten *experiment* der Klasse *BioAssay*. Das Microarrayexperiment enthält im Metafile drei weitere Informationen, die unter dem Knoten *experiment_set* wiefolgt abgekürzt werden: *Name*, *ExptSetNo* und *Description*. Jedes Experiment wird in der Stanford Microarray Database mit einem Namen (*name*) versehen, und kann im Modell der Klasse *ExperimentDesign* zugeordnet werden. Die Bezeichnung *ExptSetNo* findet im Modell keine Verwendung. Die Bezeichnung *Description* beschreibt das Experiment im Detail. Bezugnehmend auf das Modell entspricht dies dem Attribut *explanation* der Klasse *Description*. Jeder Instanz der Klasse *BioAssay* ist im Metafile ebenfalls eine Beschreibung zugeordnet,

die mit *Name* bezeichnet wird. Die Interpretation dieser Beschreibung ist im Modell mit der Klasse *ExperimentalFactor* gleichzusetzen. Die Bezeichnung *Exptid* findet keine Anwendung.

```
<experiment_set>
      !Name=Puf3delta vs. WT, SC_glycerol, Gerber AP.
      !ExptSetNo=2623
      !Description=Three independent experiments: S. cervisiae wild-type (BY4741)
                  and Puf3 mutant cells were grown in minimal medium supplemented
                  with 3% glycerol. Cells were harvested in mid-log phase by
                  centrifugation and total RNA was prepared by hot-phenol extraction.
                  cDNA was prepared with oligo-dT and using a mixture of
                  amino-allyl dUTP and dNTPs, fluorescently labeled (Cy3= wild-type,
                  Cy5 = mutant) and hybridize on S. cerevisiae DNA microarray.
            <experiment>
                  !Name=Puf3_delta vs. WT, SC_glycerol (1)
                  !Exptid=46742
            </experiment>
            <experiment>
                  !Name=Puf3_delta vs. WT, SC_glycerol (2)
                  !Exptid=47086
            </experiment>
            <experiment>
                  !Name=Puf3_delta vs. WT, SC_glycerol (3)
                  !Exptid=47301
            </experiment>
</experiment_set>
```

Abbildung 3.42: Dateiinhalt Metafile

Datafile

Das Datafile (Abb. 3.43), in dem die Scanwerte zu einem Micorarrayexperiment abgespeichert werden, ist in einer Tabellenstruktur angeordnet. In Spalte 1 sind die Gennamen aufgelistet, die der Klasse *CompositeSequence* zuzuordnen sind, in Spalte 2 die Reporterinstanzen, und ab Spalte 4 die zu einem Objekt der Klasse *BioAssay* aufgelisteten Scanwerte zu jedem Gen.

YORF	NAME	GWEIGHT	AY7n116	AY7n100	AY7n107
EWEIGHT	1	1	1		
YKR005C	1000	1	-0.32	-0.79	0.33
YKR006C	1001	1	0.74	1.05	0.45
YKR007W	1002	1	0.04	-0.53	0.22
YKR008W	1003	1	0.08	0.11	-0.17
YKL225W	1004	1	-0.04	0.43	0.25
YKR009C	1005	1	-0.42	-0.62	0.31
YKR010C	1006	1	0.27	0.22	0.11
YKR011C	1007	1	-0.38	-0.53	0.48

Abbildung 3.43: Dateiinhalt Datafile

3.10 Webbasierte Suche in der Datenbank

Die visuelle Darstellung der Datenbank *BioChipDB* wurde mit Hilfe des Tomcat-Webservers auf Basis der *JavaServer Pages*-Technologie (JSP) realisiert. Im Browserfenster der Datenbank *BioChipDB* werden alle Microarrayexperimente aufgelistet, die bisher in der Datenbank abgelegt wurden (Abb. 3.44). Pro Microarrayexperiment werden die biologischen Hintergrundinformationen aufgelistet. Dazu gehören der Titel der Publikation (*Title*), die Referenz (*Reference*), die Experimentbeschreibung (*Summary*), der Organismus (*Organism*) und die Gesamtanzahl an Versuchsreihen (*Bioassays*).

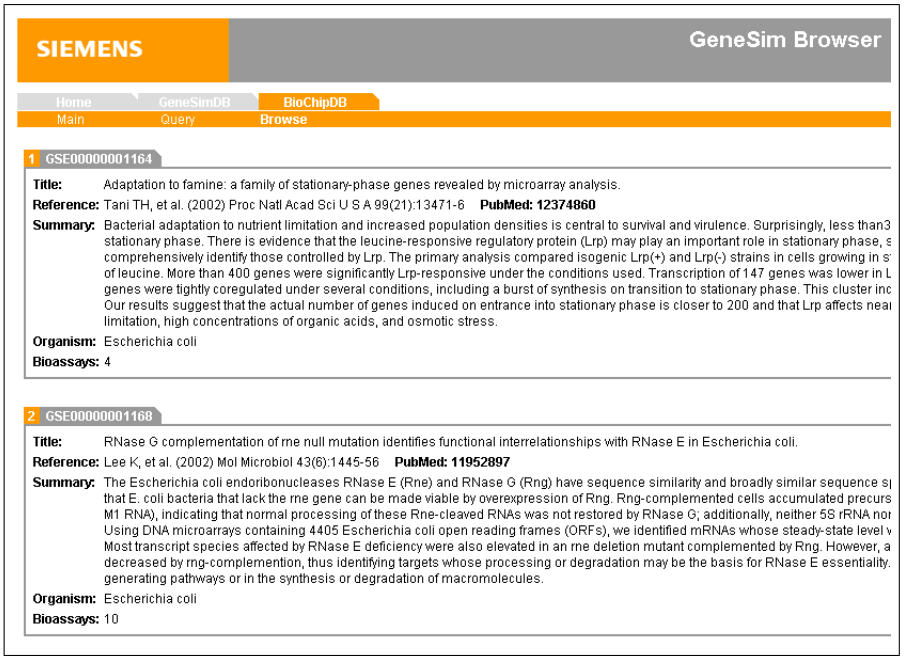

Abbildung 3.44: Webbasierte Anzeige der Microarrayexperimente

Über den Kartenreiter *Query* wird eine dynamische HTML-Seite aufgebaut, die es dem Benutzer ermöglicht, in Abhängigkeit der Spezies und einem Schlagwort in der Experimentbeschreibung ein Microarrayexperiment zu selektieren (Abb. 3.45). Dazu ist der Ogranismus und das Suchwort in die entsprechenden Eingabefelder einzutragen. Mit einem Mouseklick auf *Query*

werden die Ergebnisse in der HTML-Seite angezeigt.

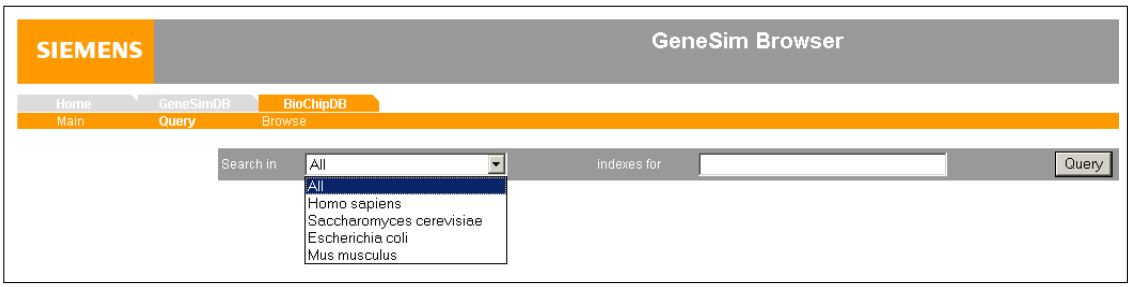

Abbildung 3.45: Suchen über Spezies und Schlagwort

Kapitel 4

Diskussion

Ziel dieser Diplomarbeit war die Entwicklung eines Datenbankdesigns zur Verwaltung von Genexpressionsdaten. Die Herausforderung bestand darin, ein Tabellendesign zu implementieren, welches sich auf ein allgemeingültiges Klassenmodell bezieht. Dazu war es notwendig das Klassendiagramm in ein E/R-Modell zu überführen, welches Grundlage für das Tabellendesign war. Genau hier wird deutlich, dass ein Objektmodell in eine relationale Datenbank nicht direkt übernommen werden kann, da in einem E/R-Modell Beziehungen aufgezeigt werden, die im Klassendiagramm nicht ersichtlich sind.

Die Implementierung eines Tabellendesigns gemäß dem Objektmodell der OMG [5] bietet den Vorteil, dass Microarraydaten von mehreren wissenschafltichen Arbeitsgruppen gemeinsam genutzt werden können. Desweiteren gestaltet sich die Vergleichbarkeit von Daten wesentlich einfacher, da sie in der Datenbank an einem zentralen Punkt abgespeichert sind. Die Fokusierung der Daten führt dazu, dass der Inhalt der Datenbank in verschiedenen Ansichten graphisch dargestellt werden kann, das gerade für die Datenanalyse von großer Bedeutung ist.

Die Genexpressionsspezifikation der OMG [5] bietet dazu ein umfassendes Modell im Umgang mit Genexpressionsdaten. Die Vorteile liegen in der Ge-

neralität und Flexibilität des Modells, sowie darin, dass alle Aspekte in der Handhabung mit Genexpressionsdaten berücksichtigt werden. Für die Umsetzung des Modells in eine proprietäre Softwareapplikation oder Datenbank können auch nur einzelne Module des Modells entnommen und implementiert werden, je nachdem welcher Art die Anforderungen entsprechen.

Mit dem Speicherprogamm *ExpLoader* ist dem Benutzer die Möglichkeit gebeben worden, Genexpressionsdatensätze in die Datenbank *BioChipDB* einspielen zu können. Anfänglich gestaltete sich das Abspeichern eines Microarrayexperiments als sehr zeitintensiv, da gemäß dem Tabellendesign in der Tabelle *compSeq_reporter* beabsichtigt war, keine Instanzen der Klassen *CompositeSequence* und *Reporter* mehrfach abzuspeichern. Das Abspeichern der Applikation *ExpLoader* konnte so optimiert werden, dass zum Einspielen eines Microarraydatensatzes für *Homo sapiens* mit 30.000 Genen pro erzeugtem Objekt der Klasse *Experiment* durchschnittlich zwischen 3 und 4 Minuten benötigt wurden.

Der Webserver bietet dem Benutzer die Möglichkeit in Abhängigkeit des biologischen Hintergrunds nach einem Microarrayexperiment in der Datenbank zu suchen, auf welchem aufbauend Modellberechnungen für Genabhängigkeiten erstellt werden können. Bisher wurden Genabhängigkeiten von Microarraydaten berechnet, die die biologische Fragestellung eines durchgeführten Microarrayexperiments nicht berücksichtigt haben.

Das Speicherprogramm und der Webserver zusammen ergeben in ihrer Kombination einen Arbeitsfluss vom Abspeichern bis zur visuellen Darstellung eines Microarrayexperiments.

Das Aufstellen der funktionellen Abhängigkeiten und die daraus hergeleitete Tabellenzerlegung in BCNF garantieren für die Datenbank ein redundanzfreies Abspeichern, was für das Weiterbestehen der Datenbank *BioChipDB* von großer Bedeutung ist.

Für die Zukunft und Perspektiven des Speicherprogramms, des Webservers

und der Datenbank sind folgende Aspekte vorzuschlagen:

- Die Implementierung eines GUI (*Graphical User Interface*) für das Speicherprogramm ermöglicht dem Benutzer ein intuitiveres Arbeiten im Umgang mit Microarraydatensätzen.

- Die vollständige Implementierung des Klassenmodells in Java gemäß dem Standard der Genexpressionsspezifikation der OMG [5], um alle möglichen Informationen zu einem Microarrayexperiment abspeichern zu können.

- Das Laden von Microarraydatensätzen aus der Datenbank *ArrayExpress* [14] und das Laden des Standardformats MAGE-ML (*Microarray genexpression markup language*), einer XML-basierten Sprache, welche Genexpressionsdaten speichert.

- Das Erweitern des Webservers um Funktionalitäten der Clusteranalyse, Abspeichermöglichkeiten eines Microarrayexperiments aus der Datenbank in den Dateityp MAGE-ML, sowie die genspezifische Suche in einem Experiment.

Das Projekt *BioChipDB* ist über diese Diplomarbeit hinaus im operativen, industriellen Einsatzbereich.

Literaturverzeichnis

[1] Alberts, B. *Lehrbuch der molekularen Zellbiologie.* WILEY-VCH Verlag GmbH, 2000.

[2] Edgar, R., Domrachev, M., Lash, A. E. Gene Expression Omnibus: NCBI gene expression and hybridization array data repository. *Oxford University Press*, 30(1):207–210, 2002.

[3] Watson, J., Crick, F. Molecular structur of nucleic acids. *Nature*, 171:737–738, 1953.

[4] Booch, G. *Object–Oriented Analysis and Design with Applications, 2nd ed.* Benjamin/Cummings, 1994.

[5] Object Management Group. *Gene Expression Specification, Version 1.1, formal/03-10-01.* www.omg.org, 2003.

[6] Balzert, H. *Lehrbuch der Software–Technik.* Spektrum Akademischer Verlag, 2000.

[7] Date, C. J. *An Introduction to Database Systems, 8th ed.* Addison-Wesley, 2004.

[8] MySQL. *MySQL Reference Manual.* www.mysql.com, 2005.

[9] Gerber, A. P., Herschlag, D., Brown, P. O. Extensive association of functionally and cytotopically related mRNAs with puf family RNA-binding proteins in yeast. *PLoS Biol*, 2(3):E79, 2004.

[10] Schena, M., Shalon, D., Davis, R. W., Brown, P. O. Quantitative monitoring of gene expression patterns with a complementary DNA microarray. *Science*, 270:467–470, 1995.

[11] Erler, T. *Das Einsteigerseminar UML.* bhv Verlag, 2000.

[12] Brazma, A. *et al.* Minimum information about a microarray experiment (MIAME) – toward standards for microarray data. *Nature Genetics*, 29:365–371, 2001.

[13] Lander, E. S. *et al.* Initital sequencing and analysis of the human genome. *Nature*, 409:860–921, 2001.

[14] Parkinson, H. *et al.* ArrayExpress – a public repository for microarray gene expression data at the EBI. *Nucleic Acids Research*, 33:D553–D555, 2005.

[15] Roberts, L. *et al.* A history of the genome project. *Science*, 291:1195, 2001.

[16] Sherlock, G. *et al.* The Stanford Microarray Database. *Nucleic Acids Research*, 29:152–155, 2001.

[17] Baldi, P., Hatfield, G. W. *DNA Microarrays and Gene Expression.* Press syndicate of University of Cambridge, Cambridge, MA, 2002.