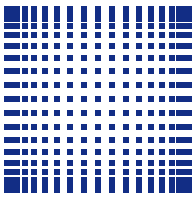
Diplomarbeit

Entwurf und Implementierung einer Datenbank in einem Analysesystem für die Vergleichende Genomik

Alfred-Wegener-Institut
für Polar- und Meeresforschung
in der Helmholtz-Gemeinschaft



hochschule mannheim

Fakultät für Informationstechnik

Ausführungszeit:	01.09.2006 – 28.02.2007
Abgabedatum:	31.03.2007
Studiengang:	Technische Informatik
Erstellt von:	Sascha Zielke
Matrikelnr.:	0121263
Projektbetreuer:	Prof. Dr. Stephan Frickenhaus, Dr. Klaus Valentin, Dr. Bánk Beszteri, Prof. Dr. Eckhart Körner

Ehrenwörtliche Erklärung

Ich versichere, dass ich die vorliegende Diplomarbeit selbständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten und nicht veröffentlichten Schriften entnommen wurden, sind als solche kenntlich gemacht. Die Arbeit hat in dieser oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegen.

Mannheim, den 30.3.2007

Sascha Zielke

Kurzfassung

Diese vorliegende Diplomarbeit entstand unter der Federführung des Alfred-Wegener-Institutes für Polar- und Meeresforschung.

Ziel der Arbeit ist es, eine Datenbankanbindung an das bereits zur phylogenetischen Analyse eingesetzte Programm PhyloGena zu entwerfen und zu implementieren. Die derzeitige Datenverwaltung von PhyloGena sieht eine Speicherung der Daten im Arbeitsspeicher vor, welche bei Beendigung des Programms verloren gehen. Dies hat in erster Linie den Nachteil, dass die Datenmenge, welche PhyloGena zur Verfügung steht durch die Größe des Arbeitsspeichers limitiert wird. Außerdem besteht keine Möglichkeit nachträglich auf einzelne Daten zuzugreifen.

Durch eine Datenbank, welche die anfallenden Daten zur Laufzeit von PhyloGena erfasst und verwaltet, sollen diese Schwächen behoben werden. Maßgeblich wird sich durch diese Maßnahme erhofft, dass phylogenetische Analysen mit einem deutlich höheren Datenvolumen durchgeführt werden können, um so eine bessere Einordnung einer unbekannten Gensequenz in den evolutionsgeschichtlichen Ablauf zu erhalten.

Vorwort

Das Alfred-Wegener-Institut wurde 1980 als Stiftung des öffentlichen Rechts gegründet [1]. Diese Stiftung umfasst neben dem Hauptsitz in Bremerhaven auch die Wattenmeerstation Sylt, die Forschungsstelle Potsdam (seit 1992) und die Biologische Anstalt Helgoland (seit 1998). Die Stiftung ist Mitglied der Hermann von Helmholtz-Gemeinschaft Deutscher Forschungszentren (HGF) und als solches Zentrum für Polar- und Meeresforschung. Um das Ziel der Forschungsarbeiten am Alfred-Wegener-Institut, die Veränderung der globalen Umwelt und des Erdsystems zu entschlüsseln, konsequent zu verfolgen, forscht das Institut in drei Fachbereichen:

- Geowissenschaften,
- Biowissenschaften und
- Klimawissenschaften.

Im Fachbereich Geowissenschaften werden die Prozesse der Erde auf die Entwicklung des Klimas untersucht. Hierbei spielen die Eiskappen der Pole, die terrestrischen Ablagerungen und der Sedimentabbau der Ozeane eine wichtige Rolle. Erforscht werden in diesem Zusammenhang u.a. die Struktur und die Veränderung der Erdkruste und der polaren Eisschilde.

Der Fachbereich Biowissenschaften bearbeitet ökologische, physiologische und ökotoxikologische Fragestellungen. Die Hauptthemen liegen hierbei bei den Reaktionen von Zellen, Organismen, Gemeinschaften und Ökosystemen.

Im Fachbereich Klimawissenschaften werden die physikalischen und chemischen Vorgänge im System Ozean-Eis-Atmosphäre und ihre Bedeutung für die weltweite Klimaentwicklung untersucht. Aktuell haben auch Mitarbeiter dieses Fachbereichs am internationalen Klimareport des IPCC mitgewirkt.

Die Arbeitsgruppe Bioinformatik am Alfred-Wegener-Institut ist an das Rechenzentrum angegliedert und umfasst neben einigen Praktikanten und Diplomanden maßgeblich zwei Mitarbeiter, welche sich mit der Datenauswertung und der Algorithmik in der Molekularbiologie beschäftigen.

Das Arbeitsgebiet ist dementsprechend äußerst vielseitig. Es reicht von der Bereitstellung großer bioinformationtechnisch nutzbarer Rechenkapazität bis

hin zur Entwicklung neuer Auswertungsmethoden und die damit verbundene Integration in bestehende Workflows.

Das Rechenzentrum deckt mit seinen Diensten einen großen Bereich ab, dieser erstreckt sich über die kompetente Unterstützung von AWI-Mitarbeitern hinsichtlich informationstechnologischer Belange bis hin zu wissenschaftlichen Hochleistungsberechnungen für die Klimaforschung, wie z.B. die Bestimmung von Tsunamis. Die technische Infrastruktur für diese Dienste besteht aus dem lokalen Rechnernetz an den jeweiligen Standorten des Alfred-Wegener-Instituts, sowie der Ankopplung dieser Netzte an das weltweite Datennetz durch Satellitenverbindung. Die benötigte Hardware wird durch Großrechner, abgesicherte Bandarchive mit Terabyte-Kapazität und Supercomputern auf dem neusten Stand der Technik gewährleistet.

Danksagung

Hiermit möchte ich mich auch bei all denjenigen bedanken, die mich bei der Bearbeitung meiner Diplomarbeit mit fachlicher und menschlicher Kompetenz unterstützt haben und mir stets mit Rat und Tat zur Seite standen.

Ich danke Herrn Prof. Dr. Eckhart Körner, der ohne Zögern bereit war, diese Diplomarbeit zu betreuen.

Mein Dank gilt auch Herrn Prof. Dr. Stephan Frickenhaus für die Bereitstellung eines Arbeitsplatzes im Alfred-Wegener-Institut und für die hilfsbereite und nette Betreuung.

Herrn Dr. Klaus Valentin möchte ich für die Vergabe des Themas dieser Diplomarbeit, die freundliche Unterstützung während des Bewerbungsverfahrens und für die Einführung in das Gebiet der Vergleichenden Genomik danken.

Ich danke Herrn Dr. Bánk Beszteri für seine Aufopferung mir die Biologie nahe zu bringen und für seine leckeren Backkünste.

Meiner Freundin danke ich für ihre unglaubliche Geduld, die sie mit mir hatte und für die seelisch-moralische Unterstützung.

Ganz besonders danke ich meinen Eltern, ohne die mein Studium nicht realisierbar gewesen wäre.

Außerdem danke ich allen Mitarbeiterinnen und Mitarbeitern des Rechenzentrums des Alfred-Wegener-Instituts für die freundliche Unterstützung, ihre Hilfsbereitschaft und das angenehme Arbeitsklima.

Inhaltsverzeichnis

Ehrenwörtliche Erklärung **2**

Kurzfassung **3**

Vorwort **4**

Danksagung **6**

Inhaltsverzeichnis **7**

Abbildungsverzeichnis **10**

Tabellenverzeichnis **12**

1 Einführung **13**

1.1 Motivation 13

1.2 Ziel der Arbeit 13

1.3 Gliederung 14

2 Biologischer Hintergrund **15**

2.1 Biologische Grundlagen 15

2.2 Genomannotation 19

2.3 Homologie 20

2.4 Phylogenetische Analyse 21

2.5 Beziehung zwischen Plastidengenen in Rotalgen und Diatomeen 22

3 Technologischer Hintergrund **23**

3.1 Analysesoftware 23

3.1.1 Sequenzdatenbanken 23

3.1.1.1 Primäre Sequenzdatenbanken 23

3.1.1.2 Sekundäre Sequenzdatenbanken 24

3.1.2 Swiss-Prot 25

3.1.3 BLAST 26

3.1.4 Multiple Alignment 31

3.1.5 Phylogenetische Bäume 37

3.1.5.1 Distanzbasierte Methoden 38

3.1.5.2 Charakterbasierte Methoden 40

3.1.6 PhyloGena 41
3.2 Relationale Datenbanken 44
3.2.1 SQL 46
3.2.2 MySQL-Datenbanken unter Java/Eclipse 47
3.2.3 Administration und Verwaltung von MySQL-Datenbanken 48

4 Konzept 50
4.1 Information der Datenwerte 50
4.2 Fachkonzept zur Datenbankanbindung 53
4.3 Tabellenentwurf zur Datenbankanbindung 55
4.4 Konzeptioneller Aufbau des Datenmodells von PhyloGena 57
4.5 Konzeptioneller Aufbau des Abfrage- und Visualisierungsmodells von PhyloGena 60

5 Realisierung 63
5.1 Datenbankimplementierung 63
5.1.1 Datentypen 63
5.1.2 Datenbankverbindung 64
5.2 Datenakquise 65
5.2.1 Tabelle „config“ 65
5.2.2 Tabelle „query“ 66
5.2.3 Tabelle „blast“ 66
5.2.4 Tabelle „blast_result“ 66
5.2.5 Tabelle „neighbor“ 67
5.2.6 Tabelle „minNeighbor“ 67
5.2.7 Tabelle „sequence“ 67
5.2.8 Tabelle „alignment“ 68
5.2.9 Tabelle „tree“ 68
5.3 Visualisierung der Daten in PhyloGena 70
5.3.1 Panels 70
5.3.2 SELECT-Abfragen 71
5.4 Vordefinierte SQL-Befehle 73
5.5 Eingabe eigener SQL-Befehle 74

6 Zusammenfassung und Ausblick 75
6.1 Zusammenfassung 75
6.2 Ausblick 76

Anhang A: CD-ROM 78

Anhang B: Literaturverzeichnis 79

Anhang C: Abkürzungsverzeichnis 83

Anhang D: Quellcode Datenbankerstellung 85

Abbildungsverzeichnis

Abbildung 2.1: DNA Molekül; Quelle [3] 16
Abbildung 2.3: Transkription-Translation; Quelle: [5]....19
Abbildung 3.1: Anzahl der Sequenzeinträge in der EMBL-Bank von 1982 bis 2007; Stand: März 2007; Quelle: [15] 24
Abbildung 3.2: Sequenzalignments zweier Proteine; Quelle: [18] 26
Abbildung 3.3: Kostenverteilung der Sequenzmatrix; Quelle: [19] 28
Abbildung 3.4: Alignmentgitter für v = ATGTTAT und w = ATCGTAC. Jedes Alignment entspricht einem Pfad in dem Alignmentgitter von (0, 0) nach (n, m) und jeder Pfad von (0, 0) nach (n, m) in dem Gitter entspricht einem Alignment; Quelle: [19] 29
Abbildung 3.5: Clusteranalyse = Erstellung eines "guide tree"; Quelle: [22] 33
Abbildung 3.6: Alignment von nahe verwandten Sequenzen; Quelle: [22] 33
Abbildung 3.7: Sequenzpaare; Quelle: [27]....35
Abbildung 3.8: Diagonale von Sequenzpaaren; Quelle: [27] 35
Abbildung 3.9: Multiples Alignment mit Dialign; Quelle: [27] 35
Abbildung 3.10: Zu alignierende Sequenzen mit POA; Quelle: [29] 36
Abbildung 3.11: POA-Darstellung von zwei Alignments; Quelle: [29]....36
Abbildung 3.12: Phylogenetischer Baum mit POA; Quelle: [29] 37
Abbildung 3.13: Phylogenetischer Baum....38
Abbildung 3.14: Begriffserläuterung für Tabellen in relationalen Datenbanken; Quelle: [33]....44
Abbildung 3.15: Beispiel zwischen der Beziehung primary key und foreign key; Der Name der Firma, primary key der einen Tabelle, wird in der Tabelle „Vorträge“ als foreign key wiederverwendet; Quelle: [34] 46
Abbildung 3.16: Grafische Oberfläche zur Verwaltung von Tabellen mit phpMyAdmin; Quelle: [12] 49
Abbildung 4.1: Fachkonzept zur Datenbankanbindung an PhyloGena 53
Abbildung 4.2: Tabellenentwurf zur Datenbankanbindung 55
Abbildung 4.3: Auszug aus dem Klassendiagramm des Datenmodells....57

Abbildung 4.4: Auszug aus dem Klassendiagramm des Abfrage- und Visualisierungsmodells 60

Abbildung 5.1: Verbindungsfenster PhyloGena - Datenbank 64

Abbildung 5.2: Konfigurationsfenster PhyloGena 65

Abbildung 5.3: Beispiel eines gewurzelten Baumes; Quelle: [40] 68

Abbildung 5.4: Visualisierung der Daten aus der Tabelle „blast“ zu einer Blastsuche. 70

Abbildung 5.5: Visualisierung der Daten aus der Tabelle „blast“ zu einer Blastsuche bei mehreren Queries. 71

Abbildung 5.6: Untermenü "SQL predefined statements" zur Durchführung von SELECT-Abfragen 73

Abbildung 5.7: Eingabefenster für einen SQL-Befehl bei Aufruf des Menüeintrags "Enter new SQL statement" 74

Abbildung 5.8: Ergebnisanzeige nach Eingabe einer eigenen SELECT-Abfrage 74

Tabellenverzeichnis

Tabelle 3.1: Sequenzmatrix; Quelle: [19] 27

Tabelle 3.2: Bewertung eines multiplen Alignments mit "sum of pairs"; Quelle: [22] 31

Tabelle 3.3: Bewertung eines multiplen Alignments mit "sum of pairs"; Quelle: [22] 32

Tabelle 3.4: Zuordnungstabelle der möglichen Dateigröße zum jeweiligen Betriebssystems; Quelle: [37] 47

1 Einführung

1.1 Motivation

In der Mikrobiologie ist es häufig notwendig einen unbekannten Query, d.h. eine Gensequenz von der man die genaue Funktionsweise nicht kennt, in einen Stammbaum einzuordnen, um nach Möglichkeit die Spezies bestimmen zu können oder Rückschlüsse auf einzelne Funktionen von RNA-Strängen ziehen zu können. Hierzu verwendet man eine Sequenzanalyse, auf die später noch näher eingegangen wird. Um einen Stammbaum bilden zu können bedarf es jedoch einer großen Anzahl von Daten, welche auf Sequenzdatenbanken abrufbar sind. Das Programm PhyloGena, welches als Analysesystem am AWI entwickelt wurde übernimmt diese Aufgabe für den Biologen. Mit dem als File übergebene Query wird ein Blastprozess auf Sequenzdatenbanken gestartet. Aus den Treffern, die hierbei resultieren, wird eine sinnvolle Vorauswahl getroffen, um das Alignment zu erstellen. Abschließend kann aus dem Alignment ein Stammbaum errechnet und dargestellt werden. Der Hauptnachteil dieses Systems stellt die Datenverwaltung dar. PhyloGena wurde als eigenständiges Programm entwickelt, welches auch die enorme Anzahl von Daten abzulegen hat, dadurch kann es bei größeren Blastabfragen schnell zu einem Cache-Overflow kommen, so dass das Programm nicht mehr in der Lage ist, die anstehenden Analysen durchzuführen.

1.2 Ziel der Arbeit

Ziel ist nun, die anstehende Datenverwaltung aus PhyloGena auszulagern und die betreffenden Daten zur Laufzeit der Berechnungen abzuspeichern, damit man auf bereits berechnete Daten sofortigen Zugriff hat und nicht auf die komplette Durchführung der Blastläufe warten muss. Das Programm PhyloGena muss dementsprechend modifiziert werden, dass es als Interface dient. Auch die Visualisierung der Daten soll aktionsorientiert gesteuert werden, d.h., dass die benötigten Daten zur Visualisierung erst dann abgerufen werden, wenn sie definitiv benötigt werden. Weiterhin soll dem Benutzer die Möglichkeit eingeräumt werden direkt über das Interface PhyloGena Abfragen auf die Datenbank durchführen zu können, einmal in Form von vordefinierten Abfragen und einmal als direkte Eingabe eines SQL-Befehs.

1.3 Gliederung

Die Arbeit gliedert sich in vier Teile,

- Überblick und Hintergrundinformationen,
- Entwurf und konzeptioneller Aufbau der Datenbank,
- Realisierung und Implementierung der Datenbank und
- Zusammenfassung der Arbeit und Ausblick auf mögliche Erweiterungen.

Der erste Teil, Überblick und Hintergrundinformationen, wird in den Kapiteln 2 und 3 abgearbeitet. In Kapitel 2 werden die biologischen Grundlagen hinsichtlich der relevanten Themen aus der Molekularbiologie erläutert. Hierzu zählen, die für diese Arbeit wichtigen Begriffe wie Polymere, Genomannotation, Sequenzierung, Homologie und phylogenetische Analyse. Außerdem wird auf die Beziehung zwischen Plastidengenen in Rotalgen und Diatomeen, als Anwendungshintergrund dieser Arbeit, im Speziellen hingewiesen.

Kapitel 3 beschäftigt sich mit Themen, die den technologischen Hintergrund dieser Arbeit darstellen. Dabei werden neben diversen Analysesoftwarepaketen auch relationale Datenbanken und die wichtigste Sprache für Abfrage- und Manipulationsoperationen auf Tabellen, SQL, erläutert.

Der zweite Abschnitt, Entwurf und konzeptioneller Aufbau der Datenbank, wird in Kapitel 4 festgehalten. Dort wird auch die Struktur der Datenbank anhand von einem Fachkonzept und dem resultierenden Tabellenentwurf grafisch aufgearbeitet. Diesbezüglich finden die einzelnen Attribute der Relationen Erwähnung und werden in ihrer Verwendung erläutert.

Kapitel 5 dient der Beschreibung der entwickelten Datenbankanbindung. Hierbei wird die Datenakquise, Abfrage der Tabelleneinträge und Visualisierung der Daten beschrieben. UML-Klassendiagramme runden den Überblick über die Datenbankentwicklung ab.

Abschließend folgt in Kapitel 6 eine Zusammenfassung der Arbeit und einen Ausblick auf mögliche Erweiterungen in folgenden Projekten.

2 Biologischer Hintergrund

2.1 Biologische Grundlagen

Dieser Abschnitt behandelt die für diese Diplomarbeit relevanten Grundlagen der Molekularbiologie und orientiert sich an „Der Experimentator: Molekularbiologie/Genomics“ [2].

Die Molekularbiologie befasst sich maßgeblich mit den Nucleinsäuren, welche in DNA und RNA unterschieden werden. Wobei die chemischen Unterschiede zwischen den beiden Nucleinsäuren relativ gering sind, beide sind Polymere, die aus jeweils vier Bausteinen zusammengesetzt sind, Desoxynucleotide im Fall der DNA und Nucleotide bei der RNA. Sie dienen als Speichermedium der Erbinformation eines Organismus.

Als Polymer bezeichnet man eine chemische Verbindung in Form einer linearen Kette ähnlicher Untereinheiten. Die Linearität macht sie hinsichtlich der Bioinformatik zu einem sehr interessanten Baustein, da man sie schnell und einfach über einen String darstellen kann.

Desoxynucleotide bestehen aus einem Basenanteil (Adenin, Cytosin, Guanin oder Thymin), einem Zuckeranteil (2`-Desoxyribose) und einem Phosphatrest. Die über Phosphatreste verknüpften Zuckermoleküle bilden das Rückgrat eines DNA-Einzelstranges, eines so genannten Polynucleotids. Die an den Zucker gebundenen Basen gehen bei Zusammenlagerung zweier Einzelstränge eine Basenpaarung ein wobei Adenin mit Thymin und Cytosin mit Guanin jeweils durch zwei bzw. drei Wasserstoffbrücken verbunden werden. Auch die RNA ist nach diesem Schema aufgebaut. Sie besteht jedoch aus Nucleotiden, die anstatt 2’-Desoxyribose den Zucker Ribose und anstelle von Thymin die Base Uracil enthalten. (s. Abb. 2.1)

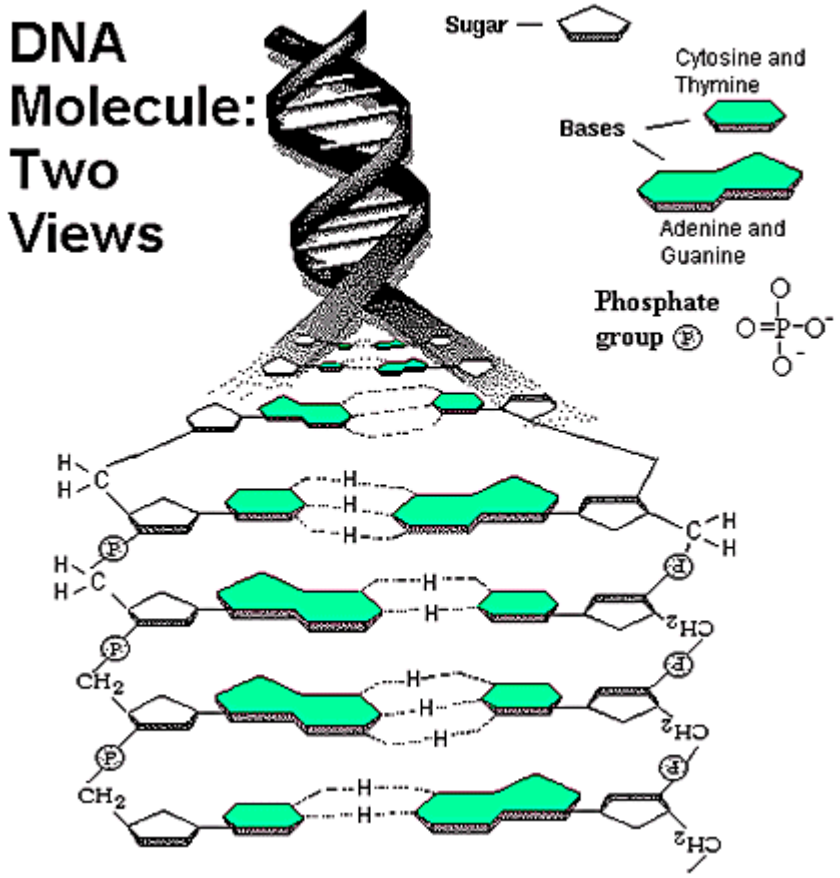


Abbildung 2.1: DNA Molekül; Quelle [3]

Um später den Begriff Sequenzierung besser zu verstehen, muss an dieser Stelle auch kurz die Biochemie Einzug erhalten.

Die Nucleoinsäuresynthese beginnt mit einem Nucleotid, an dessen 3'-OH-Gruppe eine Phosphoesterbindung mit der Phosphatgruppe des nächsten Nucleotid geknüpft wird. Somit ist für das Anhängen eines Nucleotids immer die 3'-OH-Gruppe von eminenter Bedeutung. Falls diese nicht mehr existiert, kann kein neues Nucleotid angehängt werden. Diese Erkenntnis wird sich im Falle der Sequenzierung zu Nutze gemacht. Zusätzlich zu den vier Nucleotiden werden bei der DNA-Synthese auch 2',3'-Didesoxynucleotide eingesetzt, welche an das 3'-Ende eines Polynucleotids angehängt werden können, durch das Fehlen der 3'-OH-Gruppe kann die DNA aber nicht weiter verlängert werden.

Eine Sequenz, die sich perfekt mit einer anderen Sequenz auf oben erwähnte Art (A-U bzw. A-T und C-G) paart, wird als komplementär bezeichnet. Wobei

bedacht werden muss, dass die Nucleinsäuresequenzen zweier komplementärer Nucleinsäuren keineswegs identisch, sondern eher völlig verschieden sind. Zur Verdeutlichung:

5’ – AGCTAAGACTTGTTC – 3’

3’ – TCGATTCTGAACAAG – 5’

Die Gegenläufigkeit der Orientierung 5’↔3’ resultiert aus räumlichen Gründen der Stränge.

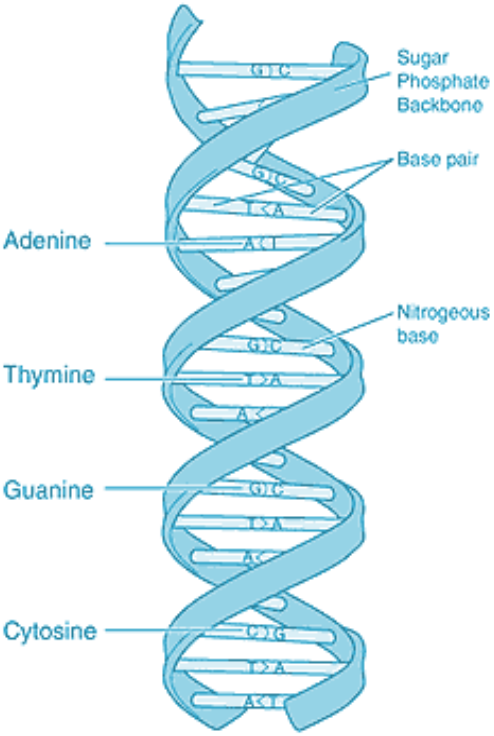


Abbildung 2.2: DNA-Strang; Quelle: [4]

In der Biologie geläufigen 5’→3’ Schreibweise sehen die beiden komplementären Stränge wie folgt aus:

5’ – AGCTAAGACTTGTTC – 3’

5’ – GAACAAGTCTTAGCT – 3’

Zwei komplementäre Stränge findet man meistens bei der DNA. RNA ist fast immer einsträngig, durch die Paarungen, welche bei der RNA auftreten erhält sie so ihre dreidimensionale Struktur. Wohingegen sich die DNA als die typische lineare Doppelhelix-Struktur (s. Abb. 2.2) darstellt. Im Labor kann diese Struktur jederzeit wieder getrennt werden. Dies wird erreicht, in dem man die Doppelhelix-Struktur zehn Sekunden einer Schmelztemperatur von 95°C aussetzt.

Auf Grund der großen chemischen Ähnlichkeit zwischen DNA und RNA ist es möglich, einen DNA-Strang in einen RNA-Strang umzuschreiben. Dieser Vorgang wird in den Zellkernen lebender Organismen von dem Enzym RNA-Polymerase durchgeführt und nennt sich Transkription. Er dient zur Generierung der messenger RNA (mRNA), die nach verschiedenen Modifikationen aus dem Kern heraus zu den Ribosomen transportiert wird. Dort wird während dem Vorgang der Translation die Sequenzinformation der mRNA in eine Amino-

säuresequenz umgeschrieben. Transkription und Translation sind demnach Teilvorgänge der Proteinbiosynthese.

Proteine sind langkettige Polymere, die aus 20 verschiedenen Aminosäuren bestehen. Während der Translation am Ribosom wird eine Aminosäure an die andere gehängt und so ein Polypeptid generiert, welches als Protein bezeichnet wird. Um nun diese 20 unterschiedlichen Aminosäuren mit nur vier Basen zu codieren, hat die Natur wohl den allerersten Code entworfen. Die Information für eine Aminosäure tragen so genannte Codons. Ein Codon besteht aus jeweils drei aufeinander folgenden Basen. Das Codon AAA steht beispielsweise für die Aminosäure Lysin, CAA für Glutamin. Daraus folgt eine Gesamtanzahl von $4^3 = 64$ Codons, wobei drei davon Stopcodons darstellen, welche das Ende einer Sequenz signalisieren. Dort endet das Polypeptid. Von der Gesamtanzahl der existierenden Codons lässt sich leicht ableiten, dass die Aminosäurekodierung redundant ist. Tatsächlich lässt sich feststellen, dass häufig nur die ersten beiden Basen über die Aminosäure entscheiden und die dritte Base ohne Bedeutung ist. So führen zum Beispiel die Codons AAA und AAG beide zur Aminosäure Lysin.

Betrachtet man die Gesamtheit der DNA, stellt man fest, dass lediglich knapp 10% der DNA-Sequenzen tatsächlich Proteine kodieren. Von den restlichen über 90% hat man noch keinerlei Kenntnisse hinsichtlich der Funktion. Die kodierenden Sequenzen werden Gene genannt. Ein Gen besteht aus einem regulaturischen und einem transkribierten Bereich. Der regulatorische Bereich kontrolliert, wann der nachfolgende Bereich transkribiert wird und wann nicht. Der transkribierte Bereich ist wiederum in zwei Arten unterteilt, in Exons und Introns. Diese Einteilung ist für das Spleißen von eminenter Wichtigkeit. Direkt nach der Synthese der transkribierte RNA wird diese von einem Molekülapparat namens Sleißosom bearbeitet. Bei diesem Vorgang werden die Introns, welche durch die so genannten „Splice Junctions“ markiert sind, aus der RNA-Sequenz herausgetrennt. Die nun übrigen Exons bilden die mRNA, die als Grundlage zur Proteinbiosynthese dient. Diese kann man nochmals in zwei Bereiche einteilen, in den Bereich, der als Informationsträger für die Synthese eines Proteins dient, den so genannten offenen Leserahmen (open reading frame, ORF) und den nicht-kodierenden Bereich, welcher als UTR (untranslated region) bezeichnet wird und dessen Funktion erst noch ergründet werden muss. Zudem besitzt die mRNA an ihrem Ende eine Sequenz von 20-100 Adenosinen, welche nicht auf der DNA kodiert sind. Dieser Poly-A-Tail signalisiert, dass die vorliegende RNA

eine mRNA ist und zur Kodierung eines Proteins dient. Die mRNA macht nur ca. 2% der RNA einer Zelle aus. Da sie aber die Sequenzinformation für die Synthese der Proteine enthält spielt sie eine eminent wichtige Rolle.

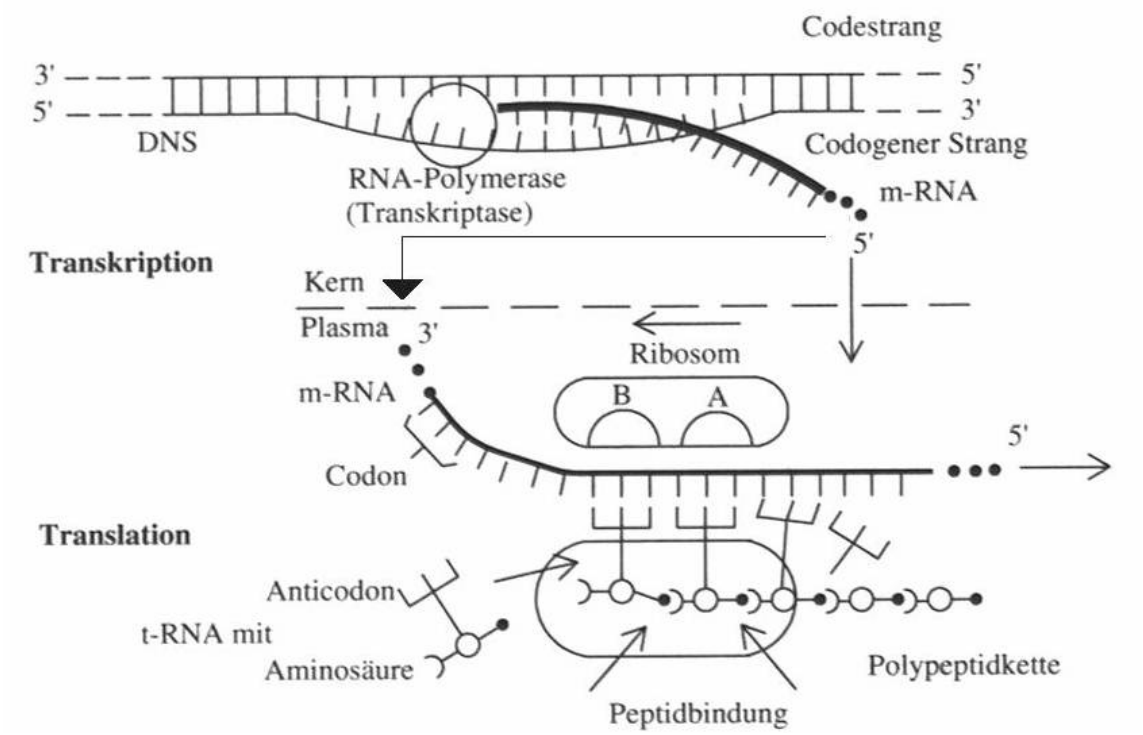


Abbildung 2.3: Transkription-Translation; Quelle: [5]

2.2 Genomannotation

Den Anwendungshintergrund dieser Diplomarbeit stellen Genomannotations-projekte. Dabei werden in der Regel die Prozesse durchlaufen, welche im Folgenden erläutert sind.

Sequenzierung Hierbei wird die DNA zuerst in kleine zusammenhängende Fragmente zerstückelt, da bei den einzelnen Sequenzierreaktionen auf Grund technischer Beschränkungen nur kurze DNA-Abschnitte von weniger als 1000 Basenpaaren abgelesen werden können [6]. Um für den Biologen eine größere Basis zu schaffen, werden diese Fragmente geklont. Anschließend werden die Fragmente sequenziert, d.h. Base für Base ausgelesen um im nächsten Schritt

am Computer wieder zu einer langen DNA-Sequenz zusammengesetzt zu werden.

Vor noch nicht einmal 20 Jahren lag der Interessensfokus der Molekularbiologen darin möglichst viele DNA-Fragmente zu sequenzieren. Heute jedoch, bedingt durch den rasanten Fortschritt in der Sequenzierungstechnologie, kommt der Molekularbiologe kaum noch mit der Auswertung der täglichen Flut an neu publizierten Sequenzen nach [7]. Um die gewaltige Anzahl von Sequenzen kontrollieren zu können, wurden spezielle Sequenzdatenbanken angelegt, auf die später noch näher eingegangen wird.

ORF-Vorhersage Um die für Gene kodierten Regionen auf der DNA-Sequenz zu finden, werden Open Reading Frames vorhergesagt. Jeder ORF beginnt mit einem Startcodon aus der Menge {ATG, GTG, TTG, CTG} und endet mit einem Stopcodon {TAG, TAA, TGA}, welches im gleichen Leserahmen liegt. Wenn nun ein ORF kodiert, wird er als Gen bezeichnet [8].

Auch für diesen Teil der Genomannotation findet man im Internet eine Vielzahl von freiverfügbarer Software für nicht-kommerzielle Anwendungen.

Annotation In diesem Schritte wird versucht aus jedem vorhergesagtem Gen die Eigenschaft des kodierten Proteins zu bestimmen. Hierfür existieren einige Analysetechniken. An dieser Stelle wird jedoch nur auf die des Sequenzvergleichs eingegangen. Dabei wird die Sequenz des ORFs mit Sequenzen von bekannten Proteinen verglichen. Kommt es hierbei zu statistisch signifikanten Ähnlichkeiten, liegt der Verdacht nahe, dass beide Proteine die gleiche Funktion besitzen.

2.3 Homologie

Bei der Annotation durch Sequenzvergleiche geht man, wie bereits erwähnt, von signifikanten Ähnlichkeiten aus. Sind zwei Sequenzen in ihrem Aufbau ähnlich, so kann man davon ausgehen, dass die Gene homolog sind, d.h. sie besitzen einen gemeinsamen Vorfahren. Daraus resultiert der Verdacht, dass die jeweiligen Proteine die gleiche Funktion erfüllen. Z. B. Hämoglobin, als Funktion hat es bei allen Wirbeltieren den Transport von Sauerstoff. [9].

In Bezug auf Gene kann man die Homologie in zwei Teilbereiche untergliedern; in die Orthologie und die Paralogie.

Orthologie Zwei Gene in unterschiedlichen Spezies, die von einem einzelnen Gen von dem direkten Vorfahr beider Spezies abstammen, nennt man ortholog. Es besteht zwar keine Gewissheit, wenn man von der Funktion des einen Gens auf die Funktion des anderen schließt, jedoch stimmen die Funktionen in den meisten Fällen überein. Es hat sozusagen eine Spezialisierung des Gens stattgefunden, aber meist nur aus dem Grund, die ursprüngliche Arbeitsweise besser ausführen zu können. [10]

Paralogie Als paralog bezeichnet man Gene, die durch Duplikation innerhalb eines Genoms entstanden sind [11]. Es sind dementsprechend exakte Kopien voneinander, auch hinsichtlich ihrer Funktion. Dies fördert die Mutationsrate des betreffenden Gens, da der Träger die Funktion des ursprünglichen Gens nicht verliert. Aus diesem Grund ist es wahrscheinlich, dass ein paraloges Gen nach einer unbestimmten Anzahl von Mutationen eine komplett andere Funktion besitzt als zu Beginn.

Aus dieser Betrachtungsweise resultiert, dass wenn man auf Basis eines Sequenzvergleichs von gleicher Funktion ausgehen will, die betreffenden Gene speziell auf Orthologie untersuchen muss. Ein Nachweis auf Homologie ist in diesem Fall nicht ausreichend.

2.4 Phylogenetische Analyse

Ein weiterer Ansatzpunkt zur Vorhersage der Funktion unbekannter Gene ist die phylogenetische Analyse. Hierbei wird ein phylogenetischer Baum erzeugt, der die Verwandtschaftsgrade von homologen Genen aufzeigt. Anders als bei dem Sequenzvergleich dienen hier nicht einfache Sequenzen als Basis, es wird vielmehr die komplette Evolutionsgeschichte einer bestimmten Gruppe homologer Gene aufgelistet und in Beziehung zueinander gesetzt; ähnlich einem Familienstammbaum.

An solch einem Baum lässt auch unterscheiden, ob ein Gen orthologer oder paraloger Herkunft ist. So lässt sich auch der Query, das unbekannt Gen, in diesen Baum einordnen, dass man Rückschlüsse auf seine evolutionäre Vergangenheit erhält und feststellen kann zu welcher Gattung er zu zählen ist.

2.5 Beziehung zwischen Plastidengenen in Rotalgen und Diatomeen

Die photosynthetischen Organellen der Pflanzen, die Plastiden, sind durch Endosymbiose entstanden. Als Endosymbiose bezeichnet man die Koexistenz eines Einzellers (hier: Mitochondrien und Chloroplasten) mit einer eukaryotischen Zelle.

Zunächst wurde ein photosynthetisches Bakterium, Cyanobakterium, von einer eukaryotischen Zelle aufgenommen. Hierbei wurde das Bakterium zu einem Plastiden reduziert und verlor ca. 90% seiner Gene an den Kern der Wirtszelle (≈ 1000 bis 2000 Gene).

Diese Endosymbiose findet man heute bei allen grünen Pflanzen und Rotalgen, man nennt sie „primäre Endosymbiose“.

Im Meer überwiegen Algen, wie z. B. die Diatomeen, deren Plastiden durch eine sogenannte „sekundäre Endosymbiose“ entstanden sind. Als „sekundäre Endosymbiose“ bezeichnet man die Situation, dass eine eukaryotische Zelle eine Rotalge mit vorhandenem Plastiden aufgenommen hat, wobei diese dann zu einem „sekundären Plastiden “ reduziert wurde. Dabei ging der Kern der aufgenommenen >Rotalge, welcher 90% der Plastidengene enthielt, verloren. Was nach wie vor unklar ist, ist ob und wie die Plastidengene aus dem verschwundenen Kern der aufgenommenen Rotalge in den Kern der neuen Wirtszelle gelangte.

Die Hypothese zu diesem Phänomen lautet, dass die aufnehmende Wirtszelle zum Zeitpunkt der sekundären Endosymbiose bereits einen primären Plastiden und die entsprechenden Gene in ihrem Kern besaß. Das würde den geringen plastidären Gentransfer aus dem Rotalgenkern in den Diatomeenkern erklären. Der erhoffte Beweis durch phylogenomische Analysen war der ausschlaggebende Punkt für diese Diplomarbeit. Eine Funktionsvorhersage von Diatomeengenen stellt aber noch eine ganz andere Herausforderung dar, denn die Genforschung hat ihr Augenmerk noch nicht so sehr auf Diatomeen gerichtet, wie man sich das an diesem Punkt wünschen würde. So existiert noch eine große evolutionäre Distanz zu Vergleichsspezies, was eine Unterscheidung zwischen paralogen und orthologen Genen schwierig macht.

3 Technologischer Hintergrund

3.1 Analysesoftware

3.1.1 Sequenzdatenbanken

Der erste Schritt einer phylogenetischen Analyse ist die Suche nach homologen Genen zu einem bestimmten Query. Hierzu stehen verschiedene Sequenzdatenbanken zur Verfügung. Die Molecular Biology Database Collection stellt eine Liste aller öffentlichen Datenbanken für die Molekularbiologie dar und wird jedes Jahr aktualisiert. In der aktuellen Ausgabe zählt sie 968 Datenbanken [12]. Sequenzdatenbanken werden zusätzlich in primäre und sekundäre Datenbanken unterschieden. Wobei die primären Datenbanken eine einfache Sammelung von Sequenzdaten repräsentieren, in welche jeder Forscher seine Ergebnisse eintragen kann. Die Einträge der sekundären Datenbanken hingegen werden von Experten überprüft und verwaltet und stellen somit ein einheitlicheres und verfeinertes Bild dar. Sequenzdatenbanken enthalten neben der eigentlichen Gensequenz noch eine Reihe anderer Einträge wie z. B. Informationen über die Herkunft eines Gens sowie Verweise auf Publikationen und andere Datenbanken. [13]

3.1.1.1 Primäre Sequenzdatenbanken

Es gibt zwei große Organisationen, welche Datenbanken für DNA-Sequenzdaten anbieten:

- European Bioinformatics Institute (EBI) in England und das
- National Center for Biotechnology Information (NCBI) in den Vereinigten Staaten.

Die EMBL-Bank des EBI umfasst in dem aktuellen Release 90 ungefähr 95 Millionen Sequenzeinträge [14]. Jedoch kann man bei Primärdatenbanken nicht davon ausgehen, dass es sich bei der Anzahl von Einträgen auch um unterschiedliche Sequenzen handelt. Da, wie bereits erwähnt, jeder Forscher Einträge vornehmen kann und sich hierbei nicht immer an die formale Struktur der Einträge hält, weisen primäre Datenbanken ein hohes Maß an Redundanz auf. Da das EBI und das NCBI sich hinsichtlich ihrer Daten mittlerweile täglich

abgleichen, ist die GenBank (Sequenzdatenbank des NCBI) als gleichwertig mit der EMBL-Bank anzusehen. Unterschiede bestehen einzig in der Formatierung der Daten.

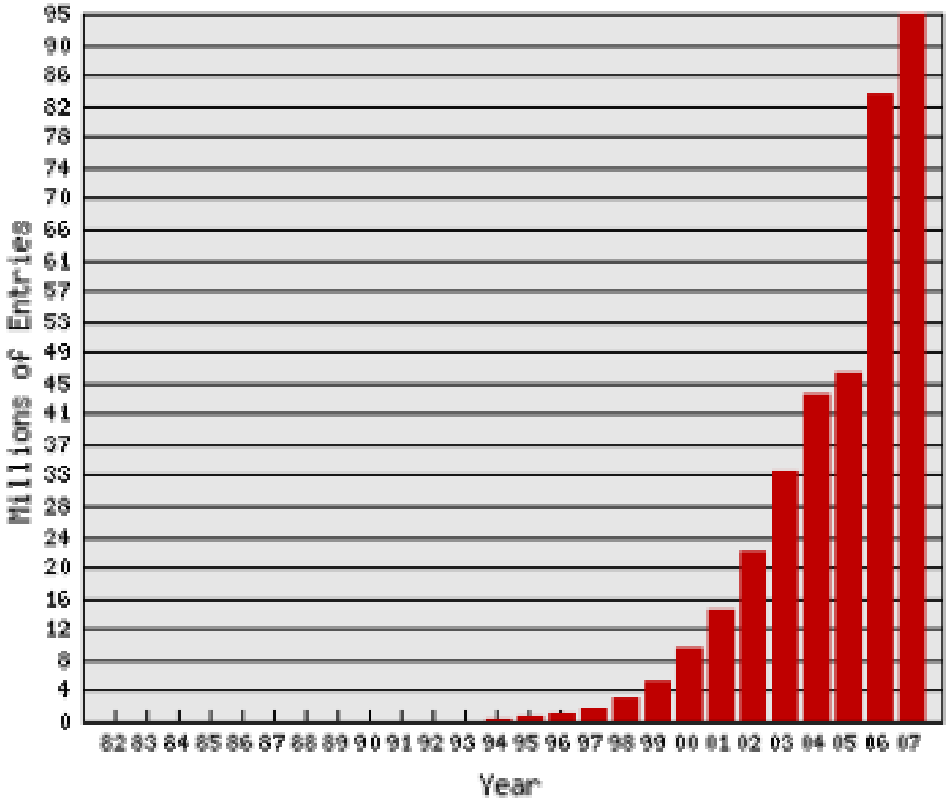


Abbildung 3.1: Anzahl der Sequenzeinträge in der EMBL-Bank von 1982 bis 2007; Stand: März 2007; Quelle: [15]

3.1.1.2 Sekundäre Sequenzdatenbanken

Auf Grund der durch reines Ansammeln von Daten enthaltenen Redundanz von Primärdatenbanken ist es notwendig geworden, sekundäre Datenbanken zu etablieren. Die Einträge basieren zwar auf denen der Primärdatenbank, werden aber von Experten oder mitunter auch von Programmen eingepflegt. Dadurch soll gewährleistet werden, dass die Datenbestände überschaubar, einheitlich und vor allem einmalig sind. Des Weiteren sind speziell die sekundären Datenbanken mit zusätzlichen Informationen zu der jeweiligen Sequenz versehen, was es wiederum dem suchenden Wissenschaftler einfacher macht, Sequenzen zuzuordnen. Der größte Nachteil von Sekundärdatenbanken und gleichzeitig auch der Grund warum Primärdatenbanken von den meisten Forschern bevorzugt werden, ist ihre Aktualität. Da die Anzahl primärer

Sequenzdatenbanken fast expotentiell anwächst, liegt häufig eine sehr große zeitliche Verzögerung zwischen der Veröffentlichung einer neuen Sequenz und deren Einpflege in die Sekundärdatenbanken.

3.1.2 Swiss-Prot

Swiss-Prot ist eine Sekundärdatenbank für Proteinsequenzen und dient mit ihren standardisierten Einträgen als Wissensbasis für Sequenzierungen. Als Sekundärdatenbank wird sie von Experten gepflegt, welche ihre Daten von diversen Primärdatenbanken beziehen. Swiss-Prot wurde 1986 gegründet und wird heute von der UniProt (Universal Protein Resource) Vereinigung gepflegt [16]. Zu dieser Vereinigung zählt

- das Swiss Institute of Bioinformatics (SIB),
- die Universität Genf,
- das bereits erwähnte EBI sowie
- das Georgetown University Medical Center for Protein Information Resource (PIR).

Das Ziel dieser Vereinigung ist es mit Swiss-Prot eine Datenbank zur Verfügung zu stellen, welche ein breites Spektrum von Annotationsinformationen zu jedem einzelnen Protein beinhaltet, dabei keinerlei Redundanz besitzt und dem Wissenschaftler oder allgemein dem Benutzer eine Vielzahl von Querverweisen auf andere Datenbanken offeriert.

Im Zusammenhang mit Swiss-Prot muss auch TrEMBL (translated EMBL) erwähnt werden. TrEMBL ist ebenfalls eine Datenbank, welche alle Übersetzungen von Nucleotideinträgen aus der EMBL-Bank enthält, welche noch nicht in Swiss-Prot integriert vorliegen. Durch die von TrEMBL computergenerierten Daten wird eine schnelle Datenbereitstellung für die Integration in Swiss-Prot gewährleistet [16]. Um auf die Daten von Swiss-Prot zugreifen zu können, gibt es unterschiedliche Wege. Der schnellste und einfachste Weg läuft über das Webinterface. Von dort können verschiedene Suchalgorithmen, SRS (Sequence Retrieval System) und BLAST (Basic Local Alignment Search Tool) gestartet werden. Ebenfalls kann man Swiss-Prot lokal installieren, dafür bietet das SIB bzw. das EBI verschiedene Formate (EMBL, XML, FASTA) zum Download an.

3.1.3 BLAST

BLAST stellt die zurzeit effizienteste Suchmaschine für homologe Sequenzen dar. Als Basis zur Sequenzsuche dienen dabei Alignments.

Als **Alignment** bezeichnet man den Vergleich von zwei oder mehr Gensequenzen. Die Vorgehensweise ist ausgesprochen simpel. Die Zeichen der jeweiligen Gensequenzen, also die einzelnen Nucleotide, werden in Zeilen übereinander notiert. Danach wird überprüft, welche Zeichen der beiden Sequenzen übereinstimmen, worauf diese mit Hilfe von Freizeichen (gaps) so verschoben werden, dass sie in einer Spalte ablesbar sind. Mit Hilfe der Alignments kann man die funktionelle oder auch evolutionäre Verwandtschaft von DNA- oder Proteinsequenzen untersuchen. Im Folgenden werden verschiedene Möglichkeiten erläutert, wie ein Alignment durchgeführt werden kann. [17]

Globales und lokales Alignment

Die Unterschiede liegen hierbei in der Länge des ausgewählten Bereichs für das Alignment. Beim globalen Alignment wird über die komplette Läge der Sequenz ein Vergleich durchgeführt. Beim lokalen Alignment werden nur Bereiche mit hoher Konservierungsrate verglichen. Daraus resultiert, dass ein lokales Alignment meist aus mehreren unterschiedlichen Abschnitten der beiden Sequenzen besteht, welche eine hohe Ähnlichkeit aufweisen. [17], [18]

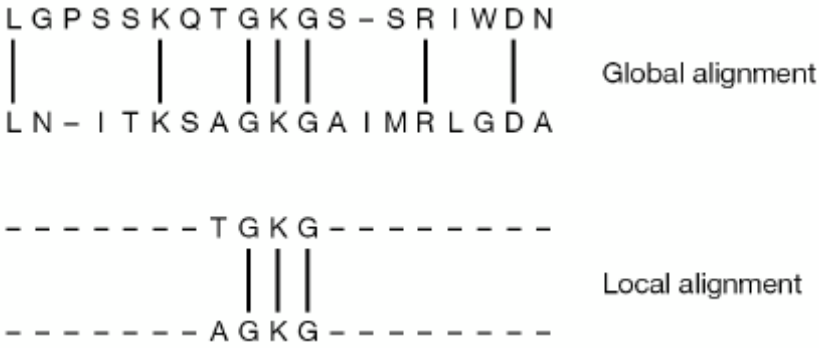


Abbildung 3.2: Sequenzalignments zweier Proteine; Quelle: [18]

Paarweises und multiples Alignment

Wenn ein Alignment aus dem Vergleich von zwei Sequenzen erstellt wurde, spricht man von paarweisem Alignment. Waren hingegen mehrere Sequenzen bei der Bildung beteiligt, liegt ein multiples Alignment vor. [17], [19]

Um mit Softwareprodukten eine sinnvolle Suche nach homologen Sequenzen durchführen zu können, benötigt man einen mathematischen Algorithmus, der eine Bewertung von Alignments vornimmt. Zwar wird von Biologen oft bemängelt, dass man ein biologisch korrektes Alignment nicht mit Hilfe eines Algorithmus berechnen kann, da mehrere unterschiedliche Aspekte in ein Alignment mit einfließen, jedoch kann ein mathematisch optimales Alignment als eine sehr gute Nährung zu dem biologischen Alignment betrachtet werden. Um eine Bewertungsfunktion für ein Alignment aufzustellen, geht man von Kosten für Substitutionen (mismatches), Löschungen (gaps) und Einfügungen (inserts) aus. Ein kleines Beispiel soll das an dieser Stelle erläutern:

Als erstes werden zwei Sequenzen (AT-GTTAT- und ATCGT-A-C) in eine m x n-Matrix geschrieben, hier eine 2 x 9-Matrix.

Tabelle 3.1: Sequenzmatrix; Quelle: [19]

A	**T**	**-**	**G**	**T**	**T**	**A**	**T**	**-**
A	**T**	**C**	**G**	**T**	**-**	**A**	**-**	**C**

Eine einfache Kostenrechnung wäre nun für

- AT-GTTAT- : 122345677 und für
- ATCGT-A-C : 123455667.

Hierbei wird lediglich die Anzahl der Zeichen einer Sequenz berücksichtigt und Freizeichen werden durch Wiederholen der vorangegangenen Zahl dargestellt. Kostenrechnungen können je nach Detailreichtum beliebig komplex gestaltet werden. Zur Anschauung soll hier aber ein simples Beispiel dienen.

Aus der Kostenverteilung ergibt sich nun folgendes Bild von Matrizen:

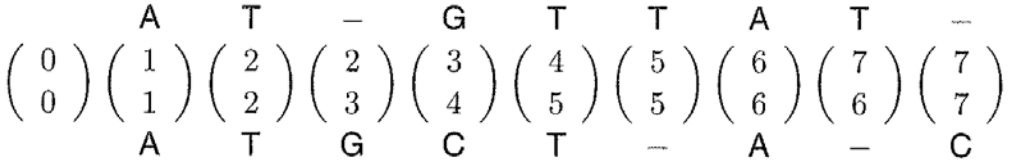


Abbildung 3.3: Kostenverteilung der Sequenzmatrix; Quelle: [19]

Um die Güte eines Alignments feststellen zu können, werden die Alignmentsequenzen in einer Matrix in Abhängigkeit ihrer Kostenverteilung gegenübergestellt, wie in Abbildung 3.3dargestellt.

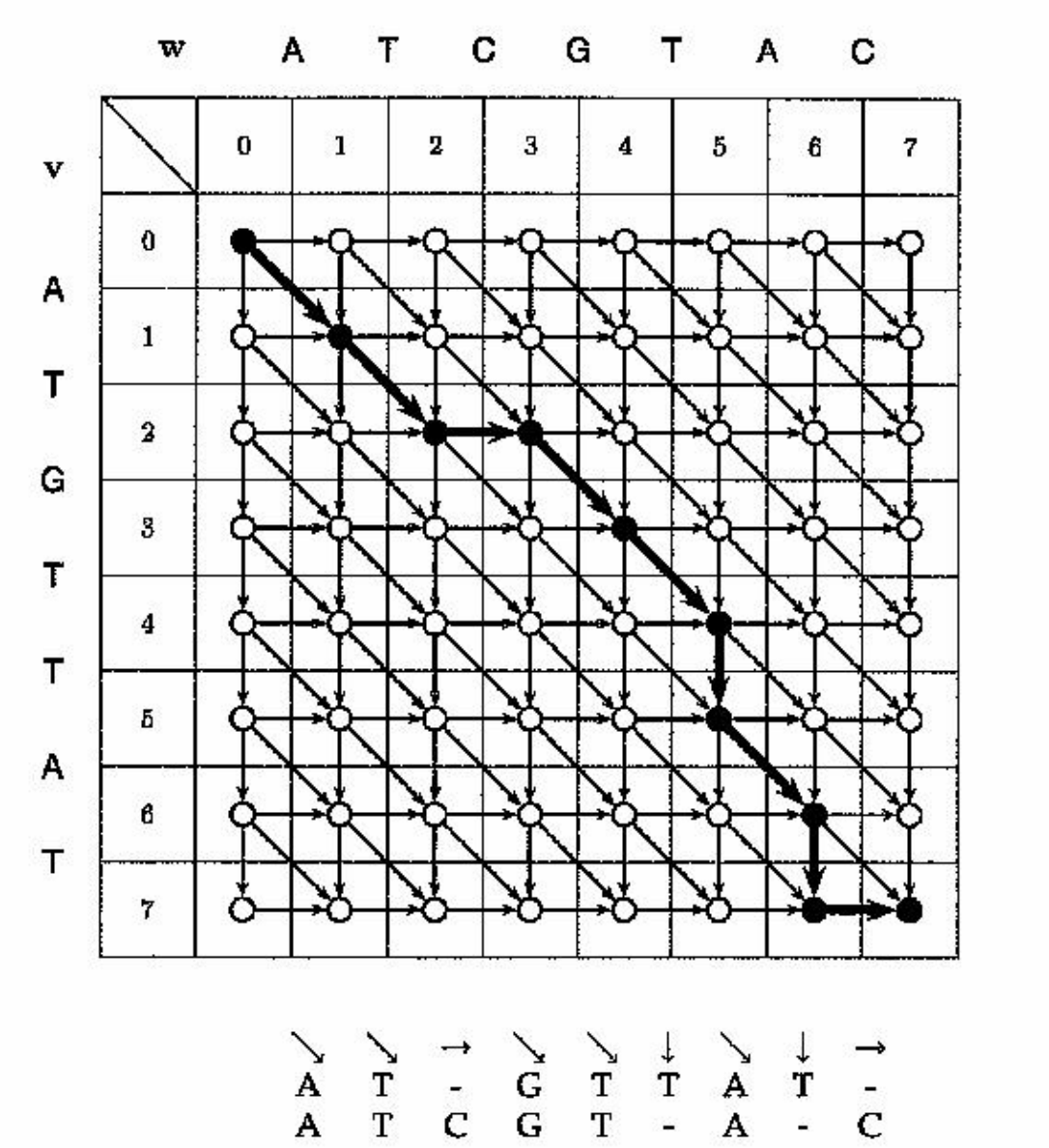


Abbildung 3.4: Alignmentgitter für v = ATGTTAT und w = ATCGTAC. Jedes Alignment entspricht einem Pfad in dem Alignmentgitter von (0, 0) nach (n, m) und jeder Pfad von (0, 0) nach (n, m) in dem Gitter entspricht einem Alignment; Quelle: [19]

Der Pfad von oben links nach unten rechts entspricht dem Alignment. Ein optimales Alignment würde absolut diagonal verlaufen und somit den kürzesten Pfad darstellen. Dies wäre aber nur dann erreicht, wenn beide Alignmentsequenzen identisch wären. Mit Hilfe von Computersoftware kann aber ein nahezu optimales Alignment berechnet werden.

Um Datenbanken, die mit einer Anzahl von Einträgen in annähernd dreistelliger Millionenhöhe aufwarten können, zu durchsuchen, ist der Ansatz mit optimalen Alignments jedoch absolut ineffizient. Aus diesem Grund wird ein heuristischer Suchalgorithmus mit deutlich geringerem Aufwand verwendet. Dabei wird die Querysequenz mit einem n Zeichen langen Fenster durchlaufen und in Abschnitte der Länge n, sogenannte n-Tupel bzw. auch Words, untergliedert. Diese Words dienen dazu eine Vorselektion der Datenbankeinträge in Hinblick auf Ähnlichkeit bezüglich des Querys vorzunehmen. Diese Menge an vorselektierten Einträgen kann dann mit Hilfe von aufwendigeren Alignmentsuchverfahren näher untersucht werden. Somit erhält man in relativ kurzer Zeit gute Vergleichswerte. [20]

Auch der **Algorithmus** von BLAST berechnet zunächst eine Liste aller Words der Querysequenz [20], [21]. Für eine Proteinsequenz nutzt er die Wortlänge n=3. Anschließend wird diese Liste mit Hilfe einer Substitutionsmatrix (BLOSUM62 oder PAM250) um genau die Words erweitert, die Ähnlichkeiten zu den bereits vorhandenen Words aufweisen. Danach startet die Suche nach den Datenbankeinträgen, die eines dieser Words enthalten. Wenn alle Datenbankeinträge gefunden wurden, wird versucht, das Alignment zu erweitern, indem die übereinstimmenden Words Zeichen für Zeichen wieder zu der ursprünglichen Sequenz ergänzt werden. So entsteht aus dem lokalen Alignment ein globales Alignment.

Jedes dieser globalen Alignments wird bewertet und auf Signifikanz untersucht. Hierzu wird der Wahrscheinlichkeitswert (p-Value) und der Erwartungswert (e-Value) jedes Alignments hinsichtlich der Anfrage errechnet. D.h. BLAST geht von einer zufällig generierten Querysequenz und der dazu gehörigen Datenbank aus und ermittelt die Werte dafür, dass in den zufälligen Daten Alignments mit gleich hoher Bewertung auftreten. Dabei wird der e-Value wie folgt berechnet:

$$E = kmne^{-\lambda S}$$

Dabei repräsentiert der Erwartungswert E die Anzahl der Alignments, die man während einer Suche in einer Sequenzdatenbank mit einem m x n großen Suchraum zufällig erhält. λS stellt eine normalisierte Austauschbewertung der verschiedenen Suchmasken und k eine Konstante dar.

Der p-Value wird mit folgender Formel berechnet:

$$P = 1\text{-}e^{-E}$$

Daraus ergibt sich, dass beide Werte für einen sinnvollen Treffer in den Datenbanken möglichst klein ausfallen sollten. BLAST reguliert dies mit einer vordefinierten Schranke. Jeder Wert, der über dieser Schranke liegt, wird bei der abschließenden Betrachtung nicht mehr berücksichtigt. Für die übrig gebliebenen Treffer wird anschließend das jeweils mathematisch optimale Alignment berechnet.

3.1.4 Multiple Alignment

Multiple Alignments bilden die Basis der phylogenetischen Analyse für Vergleiche von konservierten Bereichen in homologen Proteinsequenzen. Wie bereits in Kapitel 3.1.3 erwähnt handelt es sich bei multiplen Alignments um Vergleiche, die aus mehr als zwei Sequenzen bestehen. Dies macht sowohl die Bestimmung als auch die Bewertung deutlich komplexer.

Ein Ansatz zur **Bewertung** solcher Alignments ist die sum-of-pairs. Hierbei werden die Sequenzen wieder zeilenartig untereinander geschrieben, spaltenweise summiert, und anschließend wird die Summe der einzelnen Ergebnisse genommen, welche als Maß der Abweichung für das multiple Alignment dient. [22]

Dies soll wieder an einem Beispiel erläutert werden, wobei folgende Kosten (k) zu Grunde gelegt werden:

match k = 0; mismatch k = 2; gap k = 1

Tabelle 3.2: Bewertung eines multiplen Alignments mit "sum of pairs"; Quelle: [22]

sequence_1 =	-	G	C	T	G	A	T	A	T	A	A	C	T	
sequence_2 =	G	G	G	T	G	A	T	-	T	A	G	C	T	
sequence_3 =	A	G	C	G	G	A	-	A	C	A	C	C	T	
sum of pairs:	4	0	4	4	0	0	2	2	4	0	6	0	0	= **26**

Kleine Abweichungen können bei diesem Ansatz große Folgen nach sich ziehen, was in Tabelle 3.3 deutlich wird:

Tabelle 3.3: Bewertung eines multiplen Alignments mit "sum of pairs"; Quelle: [22]

sequence_1 =	G	-	C	T	G	A	T	A	T	A	A	C	T		
sequence_2 =	G	G	G	T	G	A	-	T	T	A	G	C	T		
sequence_3 =	A	G	C	G	G	A	-	A	C	A	C	C	T		
sum of pairs:	4	2	4	4	0	0	2	4	4	0	6	0	0	=	30

Zur **Bestimmung** von Multiplen Alignments stehen zwei Methoden zur Verfügung [17], [22]:

1) Das **Prinzip der dynamischen Programmierung** ist eine Methode, die in einer einfacheren Ausführung bereits bei den paarweisen Alignments Anwendung fand. Genügte bei dem paarweisen Alignment eine zweidimensionale Matrix, kommt bei den multiplen Alignments eine in Abhängigkeit von der Anzahl der Alignmentsequenzen n-dimensionale Matrix zum Einsatz. Algorithmen, die auf der Basis des dynamischen Programmierens agieren, wie z. B. der Smith-Waterman-Algorithmus oder der Needleman-Wunsch-Algorithmus liefern zwar optimale Alignments, sie sind aber sehr rechenintensiv und werden daher eher selten genutzt.

2) Die **heuristischen Verfahren** stellen eine Gruppe von Methoden dar, die in unterschiedlichen Ansätzen Verwendung finden. Sie erzielen ihren Vorteil durch die Geschwindigkeit der Bestimmung multipler Alignments. Im Gegensatz zur dynamischen Programmierung bieten sie jedoch nur annährend optimale Alignments. Bei den heuristischen Verfahren werden paarweise Alignments zwischen allen vorhandenen Sequenzen erstellt, woraus man alle Distanzen von jeweils zwei Sequenzen erhält. Danach folgt eine Cluster-Analyse, d.h. es wird ein Stammbaum, der so genannten Guide Tree, erstellt, bei dem ähnliche Sequenzen gruppiert werden.

	A	B	C	D
A	-	0.75	0.89	0.27
B		-	0.45	0.82
C			-	0.77
D				-

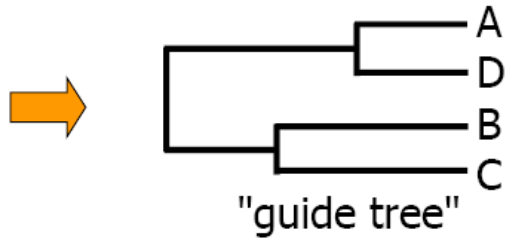


Abbildung 3.5: Clusteranalyse = Erstellung eines "guide tree"; Quelle: [22]

Anschließend werden nochmals paarweise Alignments durchgeführt, diesmal von nahe verwandten Sequenzen basierend auf dem Guide Tree, wobei die Reihenfolge mit den ähnlichsten beginnt.

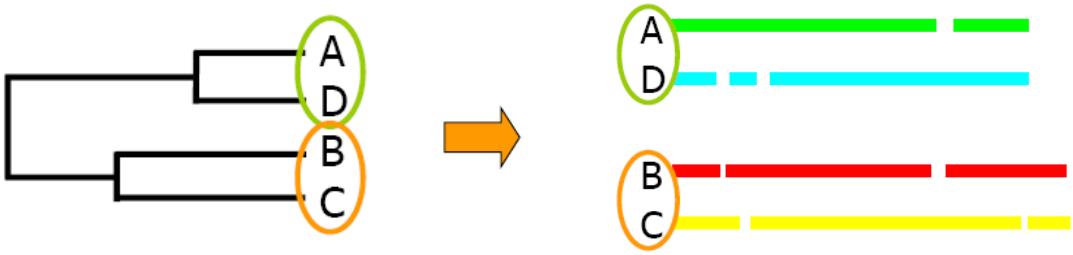


Abbildung 3.6: Alignment von nahe verwandten Sequenzen; Quelle: [22]

Abschließend folgt aus allen paarweisen Alignments sukzessiv ein globales Alignment. Der Erfolg hiervon hängt bereits von den ersten paarweisen Alignments ab. Kam es dort zu Unstimmigkeiten oder erfolgte keine klare Zuordnung, so ist auch das Endresultat unbrauchbar. Zu dem Prinzip der dynamischen Programmierung stellt es bezüglich des Resultates trotz allem eine gute Alternative dar.

Auf Grund der hohen Komplexität, die die Berechnung von multiplen Alignments bedarf, kommt man nicht umhin auf entsprechend hohe Rechenleistung zurückzugreifen. Hier seien drei Programme aufgezählt, die von größerer Bedeutung sind.

CLUSTAL CLUSTAL ist der Klassiker unter den Berechnungsprogrammen für multiple Alignments und basiert auf heuristischen Verfahren [23], [24], [25]. Zurzeit gibt es zwei frei verfügbare Versionen CLUSTAL-W und CLUSTAL-X, wobei die X Version eine grafische Benutzeroberfläche bietet. Die Qualität dieses Programms wird auch durch die Implementierung einiger kommerzieller Softwarepakete unterstrichen. Ausgehend von der Annahme, dass richtige Alignments auf evolutionäre Verwandtschaft basieren, berechnet CLUSTAL-W ein multiples Alignment mit Hilfe eines phylogenetischen Baumes (Guide Tree).

Dafür durchläuft das Programm die drei folgenden Schritte:

- Zuerst werden alle Sequenzabstände mit Hilfe der dynamischen Programmierung paarweise berechnet.
- Diese Abstände werden genutzt, um einen phylogenetischen Baum zu rekonstruieren. Dieser Baum dient der Gewichtung der einzelnen Sequenzen und bestimmt dadurch auch die Reihenfolge des resultierenden Alignments.
- Abschließend folgt das Alignment von den Blättern bis zur Wurzel des phylogenetischen Baums, wobei mit den nächstverwandten Sequenzen begonnen wird. Die Bewertung findet hier wieder in Form von Matrizen statt. Schrittweise werden immer weitere Sequenzen angefügt, indem der Mittelwert aus den bestehenden Matrixwerten zugrunde gelegt und mit der neu hinzuzufügenden Sequenz verglichen wird.

Dialign Dialign (DIagonal ALIGNment) verfolgt einen anderen Ansatz als CLUSTAL, indem eine Brücke zwischen globalem und lokalem Alignment geschlagen wird. Zur Bestimmung des Alignments werden die konservierten Bereiche von kompletten Sequenzen aligniert, wogegen nicht-konservierte Bereiche nicht berücksichtigt werden. [26], [27], [28]

Auch hier wird in drei Schritten vorgegangen:

- Im ersten Schritt wird jedes Sequenzpaar paarweise aufgetragen

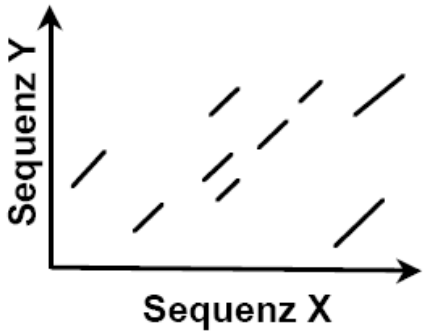


Abbildung 3.7: Sequenzpaare; Quelle: [27]

- Danach werden für jedes Sequenzpaar die Diagonalen bestimmt. Die „Diagonalen“ sind die Abschnitte, die bei einer Matrix, entstanden durch dynamisches Programmieren, als diagonal visualisiert würden (vgl. Abb. 3.4). Diesen Schritt nennt man auch „maximales Alignment“.

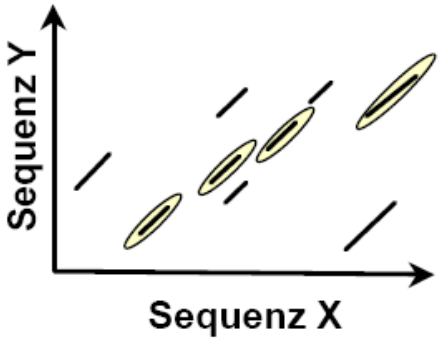


Abbildung 3.8: Diagonale von Sequenzpaaren;
Quelle: [27]

- Im letzten Schritt werden die Diagonalen aller paarweisen maximalen Alignments nach ihrem maximalen Score angeordnet und der Reihe nach in das multiple Alignment eingeführt. Zu dem entstehenden Alignment nicht-konsistente Diagonale werden entnommen.

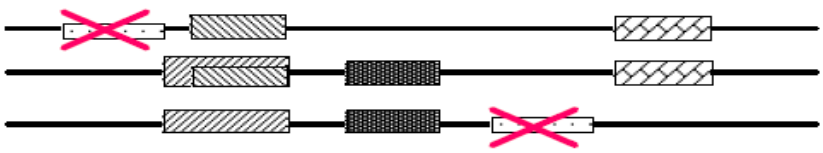


Abbildung 3.9: Multiples Alignment mit Dialign; Quelle: [27]

POA Das Programm POA (Partial Order Alignment) basiert in seinem Algorithmus auf dem heuristischen Verfahren, schließt jedoch durch seinen verbesserten Formalismus eine der größten Schwächen dieser Verfahren aus. Treten bei der Anwendung der heuristischen Verfahren zu Beginn Fehler auf, so können diese im weiteren Verlauf nicht mehr korrigiert werden. Wurden z. B. einmal zwei Sequenzen oder Alignments zu einem Alignment zusammengefasst, kann dieser Vorgang nicht mehr rückgängig gemacht werden. Die Entwickler von POA entgingen diesem Problem, indem sie sich von der Vorstellung der linearen Zeichenketten lösten. [29], [30]

Das Prinzip soll hier anhand eines Beispiels verdeutlicht werden:

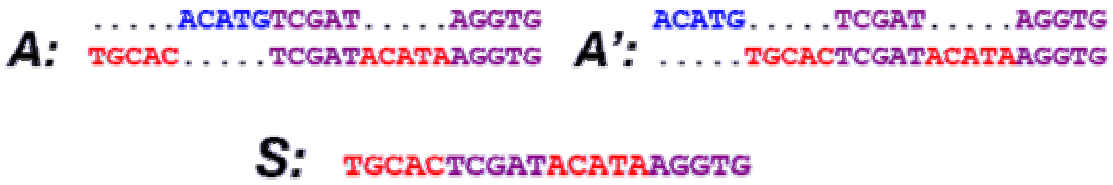


Abbildung 3.10: Zu alignierende Sequenzen mit POA; Quelle: [29]

Die Alignments A und A' sind biologisch betrachtet absolut gleichwertig. Wenn man nun eine weitere Sequenz S, welche zur zweiten Sequenz in dem Alignment identisch ist, alignieren möchte, erhält man einen Score(S, A). Dieser ist jedoch wider Erwarten nicht gleich dem Score(S, A'). Der Grund hierfür ist die Betrachtungsweise als lineare Zeichenkette. Mit Hilfe von POA lassen sich die beiden Alignments A und A' wie folgt darstellen:

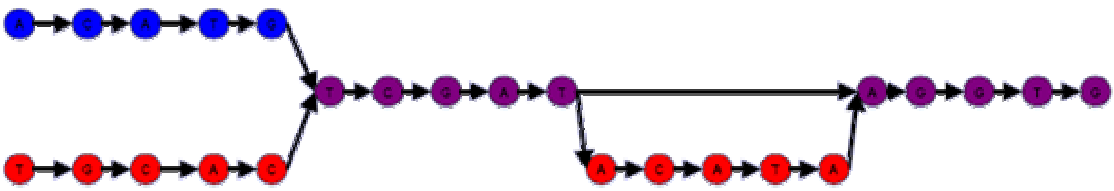


Abbildung 3.11: POA-Darstellung von zwei Alignments; Quelle: [29]

Wodurch sich die Sequenz S gleichbedeutend alignieren lässt.

Bezieht man dieses Prinzip nun auf die hierarchische Struktur von phylogenetischen Bäumen ergibt sich folgendes Bild:

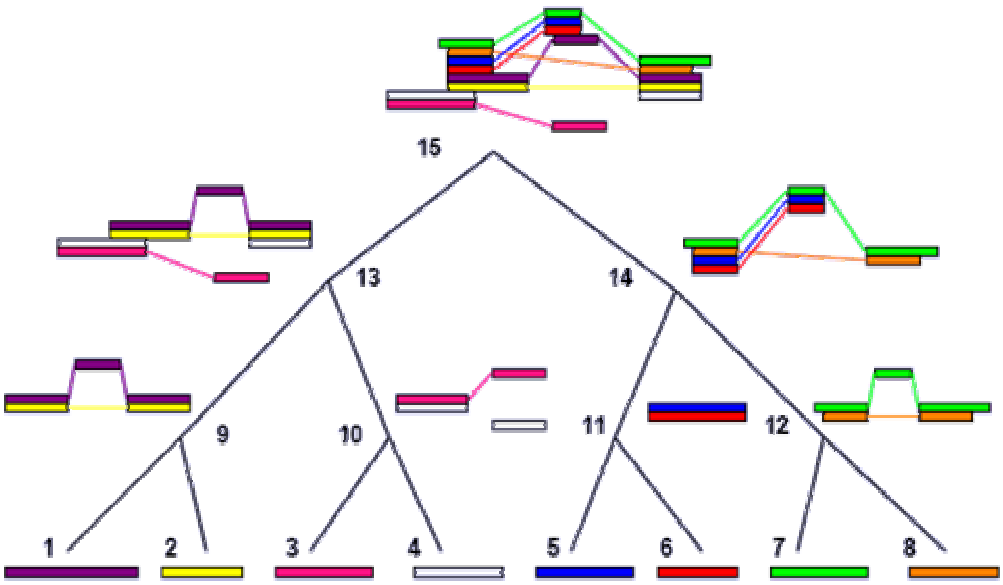


Abbildung 3.12: Phylogenetischer Baum mit POA; Quelle: [29]

Eine fixierte Position obliegt nur den alignierten Sequenzabschnitten. Nichtalignierte Sequenzabschnitte sind im weiteren Verlauf dagegen noch beliebig verschiebbar. Erreicht wurde dies unter Zuhilfenahme von gerichteten Graphen bei der dynamischen Programmierung.

3.1.5 Phylogenetische Bäume

Ein phylogenetischer Baum stellt die evolutionären Beziehungen von verschiedenen Genen, von denen man vermutet, dass sie gemeinsame Vorfahren haben, dar. In solch einem Stammbaum repräsentieren die Blätter die jeweiligen Gene, die Knoten dienen der Visualisierung von gemeinsamen Vorfahren und die Astlänge entspricht der geschätzten Zeit, in der sich ein Gen separiert hat, oder der Anzahl der Mutationen, die zu dieser Entwicklung beigetragen haben. Ist der Baum gewurzelt, so entspricht die Wurzel dem gemeinsamen Vorfahr aller sich im Baum befindlichen Gene. Ein ungewurzelter Baum hingegen kann keinen gemeinsamen Vorfahr auszeichnen und stellt lediglich die Verwandtschaftsnähe der einzelnen Gene zueinander dar. [31]

Ein Beispiel für einen phylogenetischen Baum ist in Abb. 3.13 dargeboten.

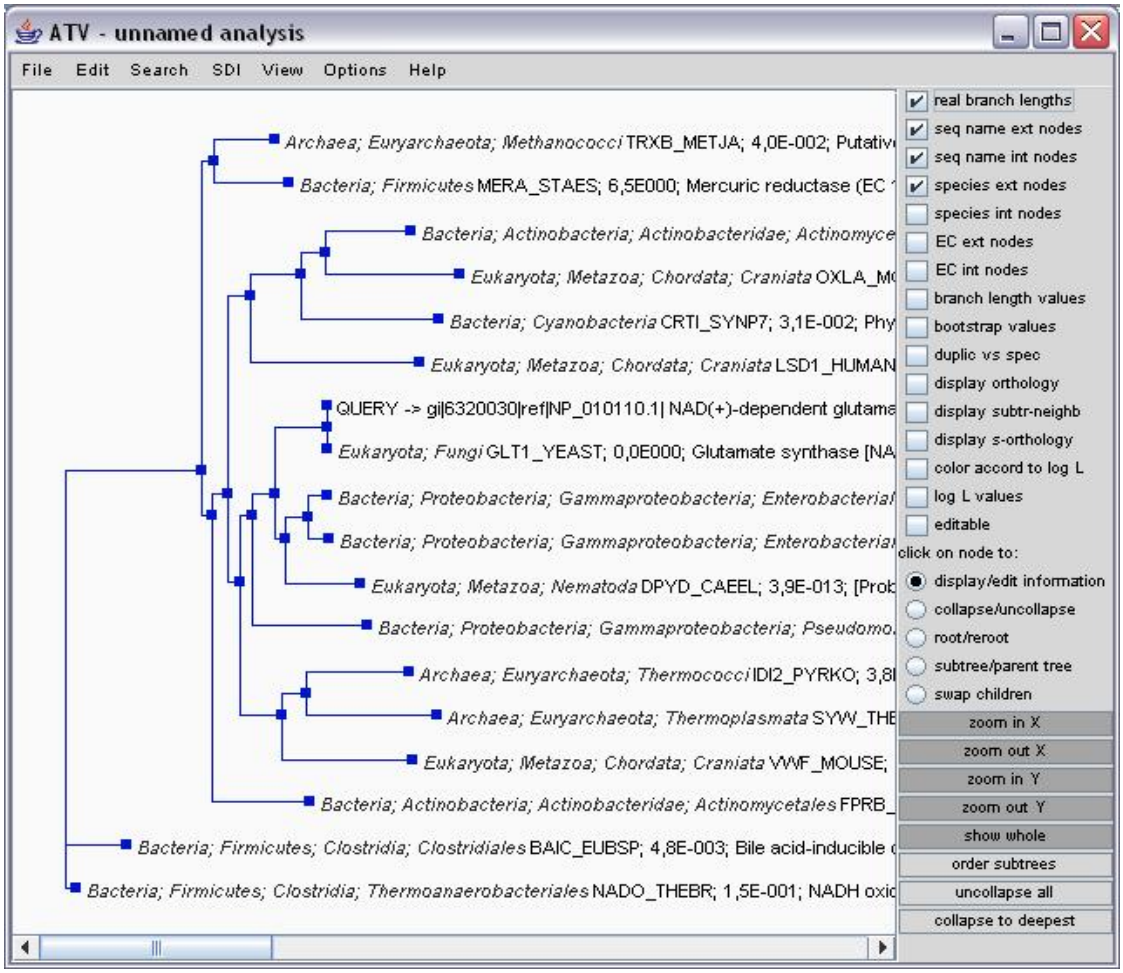


Abbildung 3.13: Phylogenetischer Baum

Zur Berechnung phylogenetischer Bäume existieren verschiedene Methoden, welche sich in zwei Oberklassen unterscheiden lassen:

Distanzbasierte und charakterbasierte Methoden.

3.1.5.1 Distanzbasierte Methoden

Zur Berechnung eines phylogenetischen Baums wird bei den distanzbasierten Methoden eine Matrix genutzt, welche die paarweisen Distanzen zwischen den jeweiligen Gensequenzen enthält. Hierzu wird ein Abweichungsmaß der Sequenzen mit Hilfe des paarweisen Alignments berechnet. Durch die Distanzmatrix kann der Baum ähnlich dem multiplen Alignment konstruiert werden. Dazu werden zuerst die nächstgelegenen Sequenzen ausgewählt und

in eine Baumstruktur gebracht. Danach werden Schritt für Schritt bzw. Sequenz für Sequenz die so entstandenen Teilbäume zu einem immer größeren Baum zusammengeführt bis alle Sequenzen abgearbeitet wurden [31]. Im Folgenden werden zwei etablierte Verfahren vorgestellt, die mit distanzbasierten Methoden zur Konstruktion phylogenetischer Bäume arbeiten.

UPGMA

UPGMA ist die Abkürzung für Unweighted Pair Group with Arithmetic Mean. Es ist ein einfaches Clusterverfahren, das bereits 1958 entwickelt wurde.

UPGMA-Algorithmus Im ersten Schritt werden zwei Sequenzen aus der Distanzmatrix bestimmt, welche die geringste Entfernung zueinander haben [31]. Sie bilden die ersten beiden Blätter des Baums und haben jeweils eine Astlänge, die der Hälfte ihrer Distanz entspricht. Sie werden über einen Knoten miteinander verbunden, woraufhin ihre Einträge in der Distanzmatrix aktualisiert werden, das bedeutet, die beiden einzelnen Sequenzen werden aus der Matrix gelöscht und neu als Gruppe abgelegt. Als Distanz der neuen Gruppe zu weiteren Sequenzen dient die mittlere Entfernung der zwei zusammengefassten Sequenzen. Im Folgenden werden nicht nur Sequenzen untereinander zusammengefasst, sondern auch Sequenzen mit Gruppen oder Gruppen mit Gruppen. Der prinzipielle Ablauf bleibt stets der gleiche und läuft solange bis nur noch eine Gruppe übrig bleibt. Dann endet der Algorithmus. [31]

Der große Nachteil dieses Algorithmus stellt seine Ultrametrik-Eigenschaft dar. Das bedeutet, alle Blätter und somit alle Sequenzen haben den gleichen evolutionären Abstand zur Wurzel. Es wird von einer molekularen Uhr ausgegangen, was bei Gensequenz noch nicht einmal theoretisch der Fall sein kann. Für Fälle, die nicht ultrametrisch sind, berechnet UPGMA falsche Bäume. [31]

Neighbor-Joining

Das Neighbor Joining Verfahren basiert auf dem gleichen grundlegenden Algorithmus wie UPGMA, jedoch mit dem entscheidenden Unterschied, dass dieses Verfahren nicht ultrametrisch ist [31]. Daraus folgt, dass phylogenetische Bäume auch dann korrekt berechnet werden, wenn die Mutationsraten von Sequenzen unterschiedlich hoch sind. Verantwortlich hierfür zeichnet sich die unterschiedliche Gruppierung der Sequenzen. Im Gegensatz zu UPGMA, bei

dem nur die kürzesten Entfernungen zur Gruppierung beitragen, bildet das Neighbor Joining Verfahren Distanzen von jeder Sequenz zu allen anderen Sequenzen, welche sich in der Distanzmatrix befinden. Auf diese Weise kann für jede Sequenz in Abhängigkeit zu einer anderen Sequenz ein Isolationsgrad bestimmt werden, der ebenfalls bei der Bildung von Gruppen berücksichtigt wird. So haben Sequenzen, deren Isolationsgrad groß ist längere Äste, da diesen Sequenzen eine höhere Mutationsrate unterstellt wird.

Der Nachteil, der bei diesem Verfahren erwähnt werden muss, ist der gegenüber dem UPGMA höhere Rechenaufwand.

3.1.5.2 Charakterbasierte Methoden

Bei den so genannten charakterbasierten Verfahren verwendet man zur Konstruktion phylogenetischer Bäume bestimmte Eigenschaften der Sequenzen, so genannte Charaktere, wobei die Kanten bzw. die Äste des Baumes mit den jeweiligen Charakteren und ihrer genauen Änderung markiert werden. Eine Kantenmarkierung indiziert, dass alle Sequenzen in dem betreffenden Teilbaum eine Änderung dieser Eigenschaft erfahren. Sie ist somit für den Grad der Verästelung eines Baumes verantwortlich. Im Bereich der charakterbasierten Methoden haben sich zwei Ansätze heraus kristallisiert, das Maximum Parsimony-Verfahren und das Maximum Likelihood-Verfahren. [31]

Maximum Parsimony

Maximum Parsimony (MP) verfolgt das Prinzip der minimalen Mutationshäufigkeit. Das bedeutet, es wird von der Annahme ausgegangen, dass die Natur keine unnötigen Mutationen hervorgebracht hat und versucht den korrekten Weg der Mutationsereignisse nachzuvollziehen, die eine heutige Sequenz in Abhängigkeit ihres Vorfahren durchlaufen haben muss. Unter all den Möglichkeiten muss nach diesem Prinzip der phylogenetische Baum der Richtige sein, der die geringsten Kantenmarkierungen und somit die geringste Anzahl von Mutationen besitzt.

Maximum Likelihood

Das Prinzip des Maximum Likelihood-Verfahrens basiert rein auf der Statistik. Bei diesem Verfahren wird für jeden Baum die Wahrscheinlichkeit berechnet,

dass aus der hierarchischen Abfolge des betreffenden Baumes die heutigen Sequenzen entstanden sind. Hierfür wird jede einzelne Position des multiplen Alignments auf seine Wahrscheinlichkeit hin untersucht. Sowohl der phylogenetische Baum als auch das dazugehörige Evolutionsmodell werden als gegeben vorausgesetzt. Die Wahrscheinlichkeit für den gefundenen Baum ergibt sich aus der Multiplikation der Einzelwahrscheinlichkeiten an den jeweiligen Positionen. Dieses Verfahren ist sicherlich eins der Interessantesten, da es absolut mathematisch begründet ist und somit den wahrscheinlichsten Rückschluss auf Verwandtschaftsbeziehungen liefert. Jedoch resultiert daraus, dass es auch das rechenintensivste Verfahren ist und für eine endliche Sequenzanzahl aus Gründen der Performance nicht mehr die gewünschten Ergebnisse erzielen kann.

3.1.6 PhyloGena

Das Programm, welches als Basis für diese Diplomarbeit fungiert, ist PhyloGena. PhyloGena ist eine am AWI entwickelte Software zur phylogenetischen Analyse. Hierzu wird im ersten Schritt ein Query im FastA-Format importiert, um anschließend einen Blastprozess auf die Sequenzdatenbanken GenBank und EMBL-Bank anzuwenden. Die gefundenen homologen Sequenzen werden dazu benutzt, ein Alignment zu berechnen und im abschließenden Schritt einen phylogentischen Baum darzustellen, anhand diesem der Biologe ablesen kann, in welcher hierarchischen Ebene sich die gesuchte Sequenz einordnen lässt. Um die Alignmentberechnung durchführen zu können, wurden die Programme

- CLUSTALW,
- Dialign und
- POA

implementiert, so dass der Biologe die Möglichkeit besitzt, unterschiedliche Berechnungsmethoden einzusetzen, um den Query richtig einzuordnen. Sie dienen folglich auch der Überprüfung des berechneten Alignments, denn sobald zwei unterschiedliche Programme dasselbe Alignment berechnen, ist die Wahrscheinlichkeit hoch, dass es sich hierbei auch um das biologisch korrekte Alignment handelt. Auch zur Bestimmung des phylogenetischen Baumes wurden unterschiedliche Softwarepakete implementiert, welche mit Hilfe der Methoden

- Neighbor Joining,
- UPGMA und
- Maximum Likelihood

zur Darstellung des Baumes beitragen. Neben der Kombination der verschiedenen Softwarepakete und deren jeweiligen Verfahren stellt die Modellierung des Vorgehens eines Biologen den größten Clou von PhyloGena dar. Das Ergebnis einer Blastsuche für einen bestimmten Query enthält in Abhängigkeit des ORFs etliche hundert Treffer. Die Aufgabe des Biologen ist es nun, aus diesen Treffern eine Auswahl zu definieren. Dies ist nötig, da Programme zur Berechnung des Alignments hinsichtlich Rechenleistung und Speicherbedarf nur mit einer begrenzten Anzahl von Sequenzen umgehen können. Außerdem dient die Vorauswahl auch der Übersichtlichkeit des im Endeffekt entstehenden Baumes. Ein Baum mit einigen hundert Sequenzen würde es dem Biologen erschweren, den evolutionären Ablauf zu rekonstruieren, da er für eine Analyse zu viel Zeit investieren müsste.

Eine Vorauswahl findet hinsichtlich zweier Kriterien statt. In erster Linie betrachtet man den berechneten E-Value. Je niedriger dieser ausfällt, desto höher ist die Wahrscheinlichkeit, dass diese Sequenzen zu der gesuchten Sequenz homolg sind. Jedoch ist für eine aussagekräftige phylogenetische Analyse eine gewisse Diversität bei der Auswahl der Sequenzen von großer Wichtigkeit. Daraus folgt, dass die Variation der Sequenzen für eine bestimmte Gruppe vielfältig sein muss, um repräsentativ zu sein. Diese Kriterien sind in ihren Ansprüchen gegensätzlich. In wie weit sie in die Vorauswahl mit einfließen, liegt daheran der subjektiven Beurteilung des Biologen.

Eine solche Beurteilung zu modellieren ist schwierig und zeichnet PhyloGena aus. Anhand von Regeln, die in Prolog verfasst wurden, wird versucht, eine Vorauswahl im Sinne eines Biologen zu treffen. Dabei ist die Selektion der Sequenzen mit den niedrigsten E-Values einfach zu modellieren. Es stellt sich jedoch maßgeblich die Frage, in wie weit die Anzahl von Sequenzen, die in das spätere Alignment mit einfließen soll, sinnvoll zu begrenzen ist. Der Ansatz, der bei PhyloGena gewählt wurde, ist prozentual definiert. D.h. die besten x Prozent der Treffer werden berücksichtigt. Diffiziler ist die Betrachtung der Diversität. Hierbei werden sowohl die Taxonomie als auch die einzelnen Gene berücksichtigt. Zuerst wird eine Selektion in Bezug auf die Spezies der jeweiligen Treffer vorgenommen. Treffer, die zu einer gleichen Spezies, z. B.

Bakterien, gehören, werden in einer Gruppe zusammengefasst. Anschließend folgt die Auswahl einer Anzahl abhängig von der Trefferrate an Sequenzen, welche die größte Homologie zu dem Query aufweisen. Diese Treffer werden als Ergebnis der Diversität der Vorauswahl zugefügt. So wird erreicht, dass alle Treffer in gewissem Maße homolog zum Query sind, aber dennoch eine Artenvielfalt repräsentieren, welche es dem Biologen durch die Analyse des phylogenetischen Baumes ermöglicht, eine genaue Zuordnung des Queries vorzunehmen.

Der Nachteil von PhyloGena ist die Verwaltung der enormen Anzahl von Daten, mit denen das Programm arbeiten muss, da für eine spätere Visualisierung alle Daten im Arbeitsspeicher gehalten werden. Für wenige Abfragen stellt das zwar kein Problem dar, will man jedoch während einer Arbeitssitzung mehrere Queries berechnen lassen, so stößt PhyloGena schnell an seine Grenzen. Um dieses Problem zu umgehen, ist eine Datenbankanbindung von Nöten, welche die Daten dann bereitstellt, wenn sie benötigt werden, um somit die Belastung von PhyloGena möglichst gering zu halten. Diese Datenbankanbindung ist das wesentliche Thema dieser Diplomarbeit.

3.2 Relationale Datenbanken

Vom prinzipiellen Aufbau kann man eine relationale Datenbank als Kollektion von Tabellen (Relationen) betrachten, in denen Datensätze abgespeichert werden. Die Datensätze bilden somit die Zeilen, auch Tupeln genannt, in einer Tabelle, wobei jeder Tupel aus einer Menge von Merkmalen, den Attributen, besteht. Jedes Attribut repräsentiert in einer Tabelle eine Spalte [32]. Das nachfolgende Bild soll die Begriffe visuell verdeutlichen.

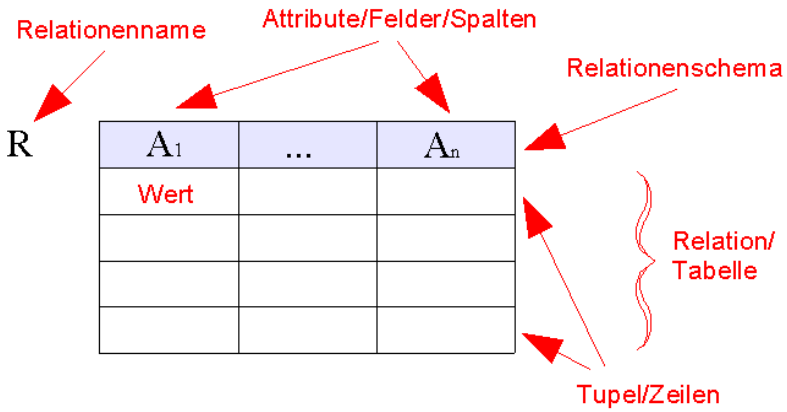


Abbildung 3.14: Begriffserläuterung für Tabellen in relationalen Datenbanken; Quelle: [33]

Um Redundanzen in Datenbanken bzw. in Relationen zu vermeiden, versucht man die Daten zu normalisieren. Redundanz repräsentiert eine unnötige Mehrfachnennung, die zu einer Erhöhung des Speicherbedarfs führen oder das Durchsuchen der Einträge zeitlich verlängern kann und ohne Informationsverlust aus der Tabelle entfernt werden kann. Aus diesem Grund wurden für Relationen so genannte Normalformen etabliert [34]:

- Erste Normalform

  Eine Relation ist in der ersten Normalform, wenn alle Attribute nur einen Wert enthalten.

- Zweite Normalform

  Eine Relation ist in der zweiten Normalform, wenn sie sich in der ersten Normalform befindet und jedes Attribut, das nicht zum Schlüssel gehört, vom gesamten Schlüssel abhängig ist.

- Dritte Normalform

  Eine Relation ist in der dritten Normalform, wenn sie sich in der zweiten Normalform befindet und es keine Abhängigkeiten zwischen Attributen gibt, die nicht zum Schlüssel gehören.

Der Sinn dieser Formalien besteht in der Minimierung von Redundanzen und dem Verhindern von Anomalien, um die Wartung einer Datenbank zu vereinfachen, eine schnelle Suche der Daten zu gewährleisten und die Konsistenz der Daten zu wahren.

In der zweiten und auch in der dritten Normalform werden so genannte Schlüssel erwähnt. Unter einem Schlüssel versteht man die eindeutige Identifikation eines Datensatzes innerhalb einer Tabelle. Sie können als einfacher Schlüssel jeweils ein Attribut beinhalten oder als kombinierter Schlüssel mehrere Attribute einer Relation umfassen.

Zur Definition eines Schlüssels sind aber zwei wichtige Schlüsseleigenschaften zu beachten [32]:

- Jeder Schlüsselwert identifiziert eindeutig einen Datensatz innerhalb der Tabelle, daraus folgt, dass verschiedene Tupel keinen identischen Schlüssel besitzen dürfen (Eindeutigkeit).
- Stellt der Schlüssel eine Kombination von Attributen dar, muss die Anzahl der Attribute minimal sein. Dies ist gewährleistet, wenn durch Streichung eines Attributes die Identifikation des Datensatzes verloren geht (Minimalität).

In der Tabellendefinition werden solche Schlüssel auch als Primärschlüssel (engl. primary key) bezeichnet. So ist für jede Tabelle in einer relationalen Datenbank laut Definition solch ein Primärschlüssel von Nöten. Weiter existieren auch so genannte Fremdschlüssel (engl. foreign key). Als solche bezeichnet man Primärschlüssel, die in einer anderen Tabelle auftreten. Somit können Identifikationsschlüssel in weiteren Tabellen wieder verwendet werden, um die gewünschte Beziehung zwischen den jeweiligen Tabellen herzustellen.

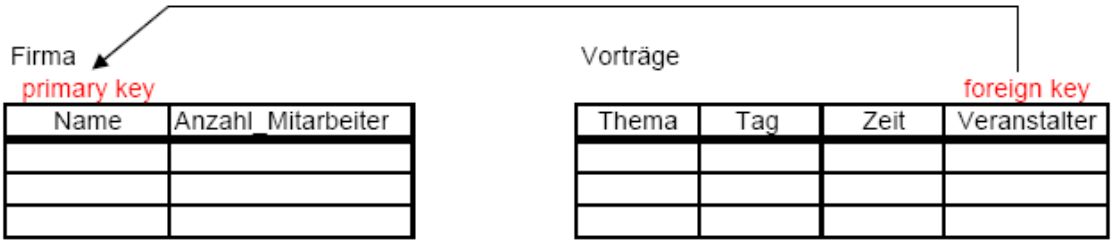


Abbildung 3.15: Beispiel zwischen der Beziehung primary key und foreign key; Der Name der Firma, primary key der einen Tabelle, wird in der Tabelle „Vorträge“ als foreign key wiederverwendet; Quelle: [34]

3.2.1 SQL

Das Relationenmodell stellt Informationen, wie im vorangegangenen Kapitel beschrieben, in Form von Tabellen dar, wobei jede Tabelle einer Menge von Tupeln bzw. Datensätzen desselben Typs entspricht. Dieses Prinzip der Mengenbildung gestattet Abfrage- und Manipulationsoperationen mengenorientiert durchzuführen. Daraus resultiert, dass jedes Ergebnis einer Abfrageoperation ebenfalls einer Menge entspricht, welche vom Datenbanksystem als Tabelle dargestellt wird. Auch eine Abfrage, die zu keinem Treffer führt bildet eine Menge, eine leere Menge, und wird demzufolge als leere Ergebnistabelle repräsentiert.

Als wichtigste Sprache für Abfrage- und Manipulationsoperation auf Tabellen ist hier SQL als Abkürzung für Structured Query Language zu nennen. SQL ist eine deskriptive, also eine beschreibende Sprache. Es genügt das Definieren der gesuchten Eigenschaften in einem SQL-Abfrage-Konstrukt, um das gewünschte Ergebnis zu erhalten. Dies wird durch die mengenorientierte Arbeitsweise von SQL ermöglicht, wobei das relationale Datenbanksystem bei einer einzigen Abfrage alle nötigen Aktionen zur Ergebnisfindung selbstständig durchführt. Somit wird der Anwender hinsichtlich der Programmierung von Suchvorgängen entlastet. Dem Anwender wird durch SQL eine einfache Syntax mit natürlich-sprachlichen Elementen dargeboten. [35], [36]

Das zurzeit wohl populärste SQL-Datenbankmanagementsystem weltweit ist MySQL. MySQL ist eine Open Source Software, welche unter zwei Lizenzen vertrieben wird, einmal einer kommerziellen Lizenz, um MySQL in kommerzielle

Applikationen einzubetten und einer freien Lizenz unter GPL (GNU General Public License). Der offene Quellcode von MySQL ermöglicht es dem Anwender die bestehende Software nach seinen eigenen Wünschen zu gestalten und zu verändern. Ein wesentlicher Vorteil von MySQL ist, dass es keine Beschränkung hinsichtlich der Anzahl von Datenbanken, Tabellen oder Datensätzen vorschreibt. Der beschränkende Faktor ist hier lediglich das jeweilige Betriebssystem. Tabelle 3.4 soll dies anhand von Beispielen für Dateigrößen in Bezug auf unterschiedliche Betriebsysteme verdeutlichen. MySQL erlaubt auch die Verwendung von verschiedenen Tabellentypen (Engines), wie z.B. das firmeneigene MyISAM, welche als Standard implementiert ist, InnoDB, durch den Aufkauf der Entwicklungsfirma InnoBase Oy eine Kooperation mit Oracle, Berkeley DB, Memory, NDB. Außerdem ermöglicht MySQL auch das Hinzufügen neuer Speicher-Engines, um beispielsweise einer Inhouse-Datenbank eine SQL-Schnittstelle bereitzustellen.

Tabelle 3.4: Zuordnungstabelle der möglichen Dateigröße zum jeweiligen Betriebssystems; Quelle: [37]

Betriebssystem	**Maximale Dateigröße**
Linux 2.2-Intel 32-bit	2 Gbyte (LFS: 4 Gbyte)
Linux 2.4+	(mit Dateisystem ext3) 4 Tbyte
Solaris 9/10	16 Tbyte
NetWare w/NSS Dateisystem	8 Tbyte
Win32 w/FAT/FAT32	2 Gbyte/4 Gbyte
Win32 w/NTFS	2 Tbyte (möglicherweise mehr)
Mac OS x w/HFS+	2 Tbyte

3.2.2 MySQL-Datenbanken unter Java/Eclipse

PhyloGena ist eine unter der Entwicklungsumgebung Eclipse in Java programmierte Software. Um Zugriff in Java auf MySQL-Datenbanken zu erhalten, ist ein sogenannter JDBC (Java Database Connectivity) Treiber erforderlich. Ein solcher Treiber stellt während dem Ablauf des Programms eine Verbindung zu den jeweiligen Datenbanken her, um anschließend Operationen auf die Relationen oder auf die Datenbank selbst anzuwenden (CREATE TABLE, SELECT, etc.). Für einen JDBC Treiber existieren unterschiedliche

Möglichkeiten der Implementierung, die in folgende vier Klassen kategorisiert werden [38]:

- **Typ 1** Treiber, die das JDBC API als Abbild eines anderen Daten APIs implementieren. Solche Treiber sind von einer bestehenden Bibliothek abhängig, was sich negativ auf ihre Portabilität auswirkt.
- **Typ 2** beinhaltet Treiber, die teilweise in Java und teilweise in Prozessor-spezifischem Code geschrieben sind. Auf Grund dieses Prozessor-spezifischen Codes sind diese Treiber ebenfalls in ihrer Portabilität begrenzt.
- **Typ 3** Treiber sind komplett in Java realisiert und der Client kommuniziert mit einem Middleware Server über ein datenbankabhängiges Protokoll. Der Middleware Server übermittelt abschließend die Datenbankabfragen an den jeweiligen Client.
- **Typ 4** Treiber sind ebenfalls komplett in Java geschrieben und implementieren das Netzwerkprotokoll für eine bestimmte Datenquelle. Somit wird gewährleistet, dass der Client direkt mit der Datenquelle kommuniziert.

Der JDBC-Treiber von MySQL, genannt MySQL Connector/J, ist ein Treiber vom Typ 4. Er kommuniziert dementsprechend direkt über das MySQL-Protokoll mit dem MySQL-Server.

Um diesen Konnektor zu benutzen, importiert man ihn zuerst in die Entwicklungsumgebung, in diesem Fall Eclipse. Anschließend kann man über die nun definierte Schnittstelle eine Verbindung zu der gewünschten Datenbank aufbauen. Mit Hilfe von Statements lassen sich über die bestehende Verbindung SQL-Operation aus dem Javaprogramm auf die entsprechende Datenbank ausführen. Somit lassen sich Datenbankentwürfe (CREATE TABLE) realisieren oder Abfragen (SELECT) durchführen.

3.2.3 Administration und Verwaltung von MySQL-Datenbanken

Zur Verwaltung von MySQL-Datenbanken bietet MySQL in einem firmeneigenen Softwarebundle den MySQL Administrator an. Dieses Softwarebundle enthält weiterhin ein Programm zur Visualisierung von MySQL-Datenbanken, den MySQL Query Browser. Mit diesem Programm ist es auch möglich,

SELECT-Abfragen durchzuführen und sich das Ergebnis relational darstellen zu lassen. Dieses Tool dient maßgeblich der Kontrolle über die Inhalte der Tabellen und über die Ergebnisse verschiedener Abfragen.

Somit stellt es ein sehr hilfreiches Testtool dar, mit dessen Hilfe Fehler in den Tabellen oder Abfragen schnell gefunden werden können.

Eine gelungene Alternative zu dem MySQL Administrator stellt phpMyAdmin dar. Bei phpMyAdmin handelt es sich um ein webbasiertes Administrationstool, mit dem man MySQL-Datenbanken über das http-Protokoll mit einem Browser verwalten kann. Der Vorteil darin besteht, dass man in der Lage ist von jedem internetfähigen Rechner aus alle Datenbanken, über die man eine Zugangsberechtigung besitzt, administrieren kann. Im Extremfall hat man so z. B. auch von einem Rechner eines Internetcafés des Ferienortes Zugriff auf die jeweiligen Datenbanken.

Die grafische Oberfläche zur Verwaltung der jeweiligen Relationen einer Datenbank ist in Abbildung 3.15 auf der folgenden Seite präsentiert. Mit Hilfe dieser Oberfläche erhält man einen Überblick der existierenden Tabellen und kann diese über vordefinierte Aktionen oder auch durch direkte Eingabe von SQL-Operationen verwalten bzw. manipulieren.

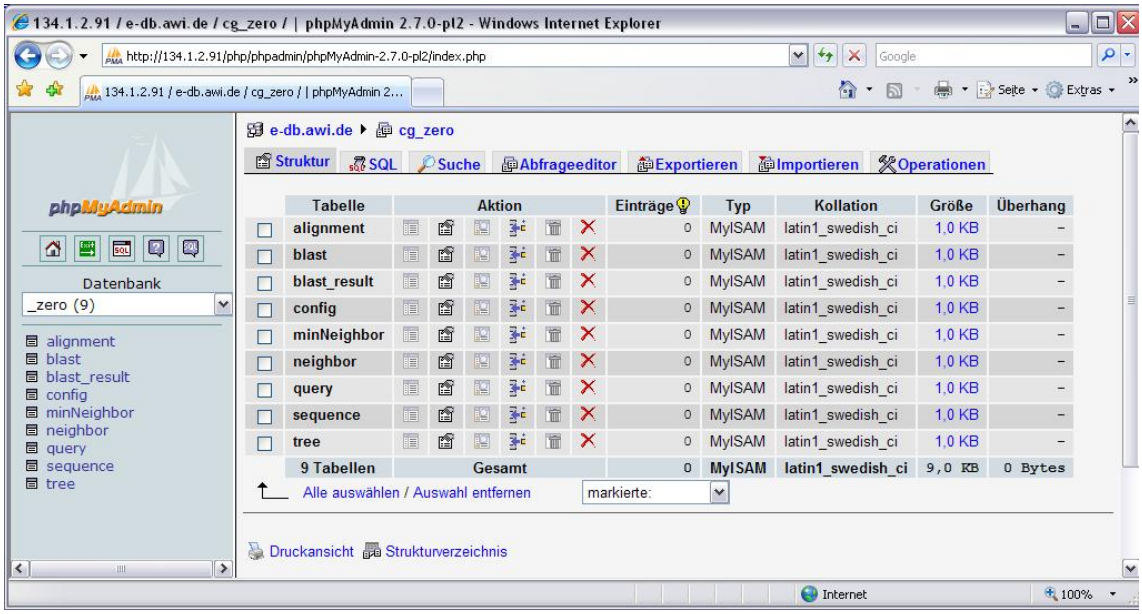


Abbildung 3.16: Grafische Oberfläche zur Verwaltung von Tabellen mit phpMyAdmin; Quelle: [12]

4 Konzept

Dieses Kapitel beschreibt den theoretischen Aufbau der Arbeit. Als Hauptschwerpunkte wird hierbei auf das Fachkonzept zur Datenbankanbindung und auf den dazugehörigen Tabellenentwurf eingegangen. Weiterhin wird an dieser Stelle auch eine konzeptionelle Darstellung von PhyloGena stattfinden.

4.1 Information der Datenwerte

Eine phylogenetische Analyse stellt eine sehr rechenintesive Operation dar. Um die Informationen, welche man durch eine solche Berechnung gewonnen hat zu archivieren, um bei Bedarf wieder auf die Daten zugreifen zu können, ist eine Datenbankanbindung an PhyloGena unumgänglich. Außerdem soll mit der Datenbankanbindung eine Leistungssteigerung hinsichtlich der Bearbeitung von mehr als 200 Blastabfragen erreicht werden. Die Daten werden hierzu zur Laufzeit von PhyloGena unter folgenden Attributen abgelegt:

- **evalueCutoff**, beinhaltet den größten noch akzeptierten Erwartungswert.
- **alignMethod**, hält die Information, mit welchem Verfahren das Alignment bestimmt wird.
- **blastRule**, hinterlegt die Regel, auf welche die Blastsuche angewandt wird.
- **selectionRule**, beinhaltet die Vorschrift, die zur Auswahl der Sequenzen für das spätere Alignment eingehalten wird.
- **treeMethode**, beschreibt die Methode, mit der abschließend der phylogenetische Baum berechnet wird.
- **maxSequences**, hält die maximale Anzahl der für das Alignment berücksichtigten Sequenzen.
- **analysisName**, liefert den selbst definierten Namen der jeweilig durchgeführten Analyse
- **blastID**, identifiziert jedes einzelne Blastergebnis in der Datenbank anhand einer Kennnummer.

- **blastVersion**, dient zur Identifikation eines identischen Blastprozesses, mit unterschiedlichen Konfigurationen.
- **selected** signalisiert, dass eine Gensequenz zur Bildung des Alignments beiträgt oder nicht.
- **geneID**, hält den Namen des jeweiligen Gens, welches durch die Blastsuche gefunden wurde.
- **queryName**, identifiziert den Query und auch den jeweiligen Blastlauf anhand eines vom Biologen vergebenen Namens. Dies macht es für den Biologen einfacher, den gewünschten Blastlauf zu lokalisieren, falls eine Eindeutigkeit der Namen gegeben ist.
- **querySeq**, beinhaltet die RNA-Sequenz des Queries.
- **eValue**, hinterlegt den Erwartungswert der gefunden Sequenz hinsichtlich einer Homologie zu dem Query.
- **species**, beschreibt die Taxonomie der jeweiligen Gensequenz.
- **description**, liefert eine Beschreibung einer Sequenz in Bezug auf ihre Entstehungsgeschichte.
- **geneName**, identifiziert die Gattung des durch die Blastsuche gefundenen Gens.
- **queryStrand**
- **queryStart**, signalisiert den Startpunkt des Abschnittes auf der RNA des Queries, der mit einem Abschnitt der dazugehörenden Gensequenz übereinstimmt.
- **queryEnd**, bestimmt den Endpunkt, des mit queryStart begonnen Abschnitts.
- **subjectStart**, lokalisiert den Startpunkt des RNA-Teilstückes der jeweiligen Sequenz, der mit einem Teilstück der Query-RNA identisch ist.
- **subjectEnd**, bezeichnet den Endpunkt, des mit subjectStart eingeleitetem Teilstückes.
- **distance**, beinhaltet die Information der evolutionären Distanz einer Gensequenz zu dem gesuchten Query.

- **minGeneID**, hält den Namen des Gens, welches die geringste Distanz innerhalb eines Blastergebnisses zum Query besitzt.
- **minDistance**, identifiziert die Distanz des unter minGeneID abgelegten Gens zum Query.
- **seqNo**, repräsentiert den Namen einer Gensequenz innerhalb eines Bearbeitungsprozesses von PhyloGena.
- **seqID**, speichert den zu seqNo gehörenden realen Namen der Gensequenz.
- **nhCode**, hält den Code zur Generierung einer Baumstruktur zur Visualisierung des phylogenetischen Baums.

4.2 Fachkonzept zur Datenbankanbindung

Die Datenbank ist in verschieden Klassen unterteilt, welche für sich selbst betrachtet eine logische Einheit bilden. Als gemeinsames Identifikationsmerkmal dienen hierbei die blastID und die dazugehörige blastVersion. Anhand dieser numerischen Merkmale lässt sich ein bestimmter Informationswert allen anderen Klassen zuordnen und eine Schnittmenge der Daten bilden.

Als Ausgangspunkt aller Klassen fungiert die Klasse config, da zu Beginn jeder Blastsuche die Konfigurationsparameter gewählt werden müssen und alle Blastresultate des gesuchten Queries in Abhängigkeit dieser Parameter stehen. Die folgende Abbildung soll dies verdeutlichen:

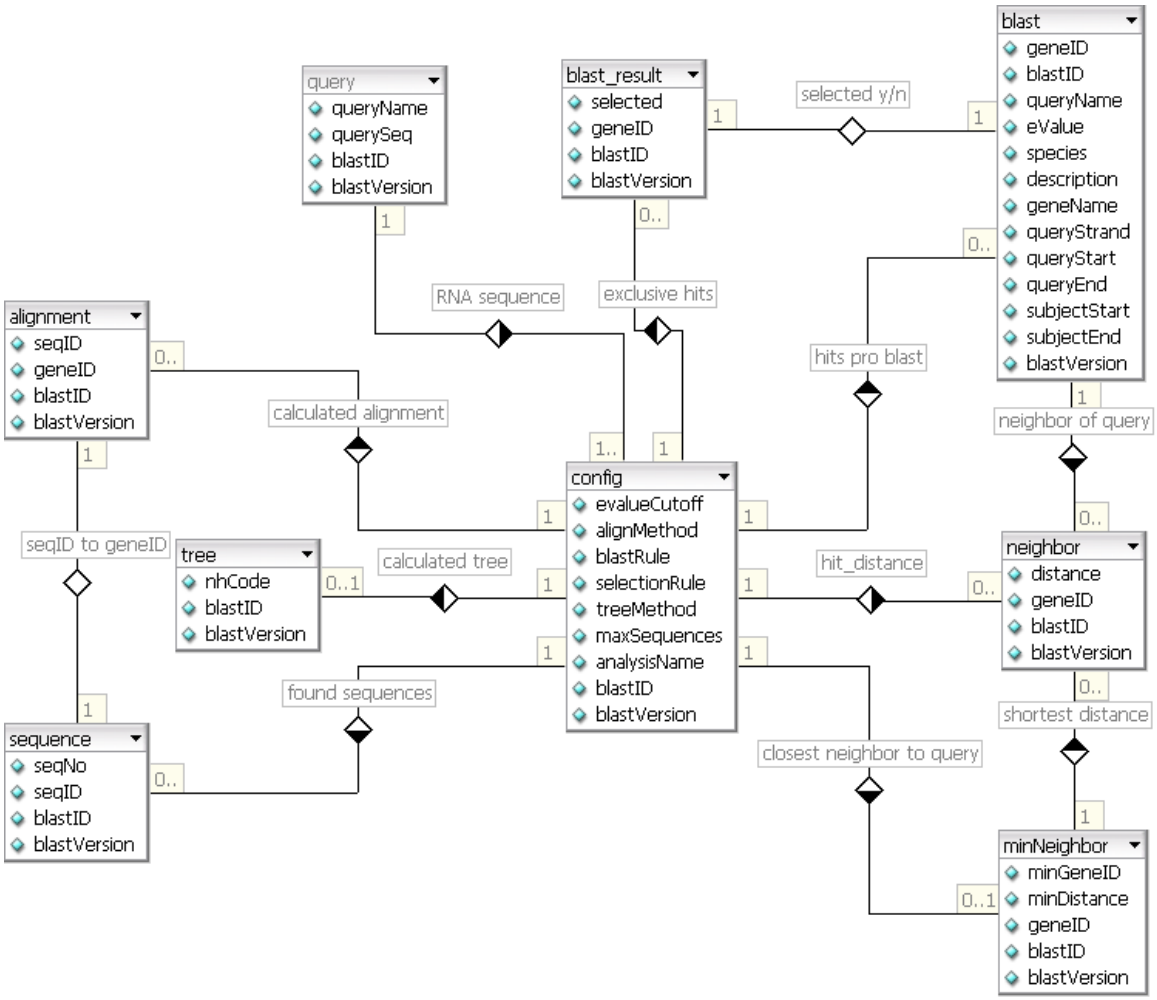


Abbildung 4.1: Fachkonzept zur Datenbankanbindung an PhyloGena

Anhand der Abbildung 4.1 lassen sich auch die Kardinalitäten der einzelnen Klassen zueinander erkennen. Hierbei nimmt die Klasse „query“ eine Sonderstellung ein, da zu einem Objekt dieser Klasse mehrere Möglichkeiten der Konfigurationsparameter existieren können. Sie bildet demzufolge eine 1:n - Beziehung zu der Klasse „config“, wobei der Wert für n mindestens 1 ist, da die Konfigurationsparameter die Basis für einen Blastprozess bilden.

Alle Objekte der restlichen Klassen beziehen sich auf das jeweilige Objekt der Klasse „config“. Daraus resultiert in den meisten Fällen eine 1:n - Beziehung, mit einem Wert $n \geq 0$, in der Komplexität der entsprechenden Klassen. Eine solche Beziehung zwischen zwei Klassen ist gegeben, falls eine Entität des einen Entitätstyps mit beliebig vielen Entitäten des anderen Entitätstyps in Beziehung steht. Diese Beziehung schließt auch den Fall mit ein, dass ein Blastprozess keine Treffer generieren konnte und somit keine Objekte der betreffenden Klasse mit Objekten der Klasse „config“ in Zusammenhang stehen.

Eine 1:1 Kardinalität indiziert, dass die involvierten Klassen eine Ergänzung der Information der einzelnen Objekte repräsentieren. Die Klasse „blast_result“ ergänzt beispielsweise die Objekte der Klasse „blast“ um die Information, ob die jeweiligen Objekte zur Bildung des Alignments berücksichtigt werden oder nicht.

Kardinalitäten, welche eine 1:0,1 - Beziehung zwischen den betreffenden Klassen aufweisen, beinhalten die Möglichkeit, dass eine Blastsuche keine Treffer lokalisieren konnte. Eine solche Kardinalität findet sich zwischen den Klassen „config“ und „tree“. Sie verdeutlicht, dass zu einem Objekt der Klasse „config“ auch kein aber maximal ein Baum gefunden werden kann. Ein Objekt der Klasse „tree“ setzt voraus, dass der Blastprozess übereinstimmende Sequenzabschnitte gefunden hat und aus den Alignments einen phylogenetischen Baum bestimmen konnte.

4.3 Tabellenentwurf zur Datenbankanbindung

Aus dem vorangegangenen Fachkonzept entstand der nachfolgende Tabellenentwurf:

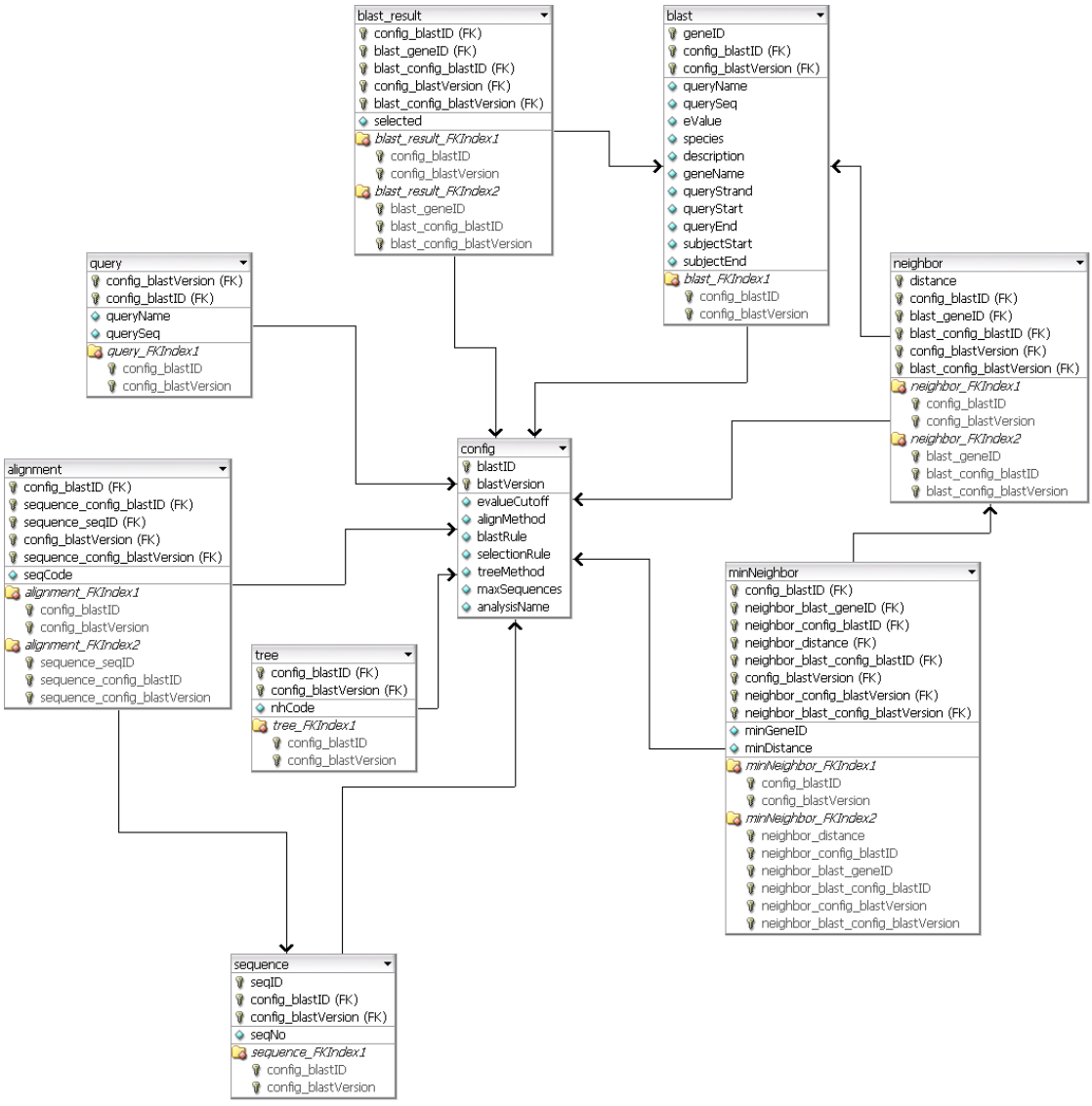


Abbildung 4.2: Tabellenentwurf zur Datenbankanbindung

Da jede Tabelle die Attribute „blastID“ und „blastVersion“ enthalten muss, um die Eindeutigkeit der Blastergebnisse zu gewährleisten, werden diese numerischen Werte noch vor dem Blastprozess generiert und in der Tabelle

„config“ hinterlegt. Sie fungieren innerhalb der Datenbank als Primary Key dieser Tabelle, welche dadurch mit jeder anderen Relation verknüpft ist.

Die Datenwerte des Attributes „geneID“ sind Kürzel der jeweils durch den Blastprozess lokalisierten Gensequenzen. Sie wird in verschiedenen Tabellen zur Bestimmung eines einzelnen Gens aus einem Abfrageergebnis genutzt. Hierbei ist es wichtig, dass innerhalb einer Blastabfrage keine Mehrdeutigkeit bezogen auf das Attribut „geneID“ existieren kann. Um dies zu garantieren, bildet die Spalte „geneID“ mit den Attributen „blastID“ und „blastVersion“ den zusammengesetzten primären Schlüssel der Tabelle „blast“.

Einen Biologen interessieren bei der Einordnung eines Queries maßgeblich die Nachbarschaftsverhältnisse zu anderen Gensequenzen und dabei im Wesentlichen die Distanz, welche zwischen dem Query und der gefundenen Sequenz liegt. Hierzu existiert zum einen eine Tabelle, welche alle Nachbarn bzw. deren Distanzen zum Query hält, und zum anderen eine Tabelle, die lediglich den Nachbarn mit seiner Entfernung ablegt, der die geringste Distanz zum Query besitzt. Aus Sicherheitsgründen besitzt die Tabelle „neighbor“ einen zusammengesetzten primären Schlüssel aus den Attributen „distance“, „geneID“, „blastID“ und „blastVersion“. Die Spalte „distance“ aus diesem Schlüssel wird in der Tabelle „minNeighbor“ als Fremdschlüssel abgefragt, so dass es nicht möglich ist, eine Distanz in der Tabelle „minNeighbor“ abzulegen, die nicht in der Relation „neighbor“ existiert.

Ähnlich verhält es sich bei den Tabellen „sequence“ und „alignment“ in Bezug auf das Attribut „seqID“. Der jeweilige Datenwert der Spalte „seqID“ identifiziert den Namen einer Gensequenz, welche während dem Ablauf der Berechnung durch PhyloGena einen numerischen Wert zugewiesen bekommt. Die Relation „sequence“ stellt eine Zuweisung des numerischen Wertes zu dem eigentlichen Namen der Gensequenz dar. Dieser Name wird durch die Tabelle „alignment“ wieder aufgegriffen. Demzufolge muss er eindeutig identifizierbar sein, was wiederum mit einem zusammengesetzten Schlüssel der Attribute „seqID“, „blastID“ und „blastVersion“ erreicht wird.

4.4 Konzeptioneller Aufbau des Datenmodells von PhyloGena

Die folgende Abbildung repräsentiert die Klassen des Datenmodells von PhyloGena:

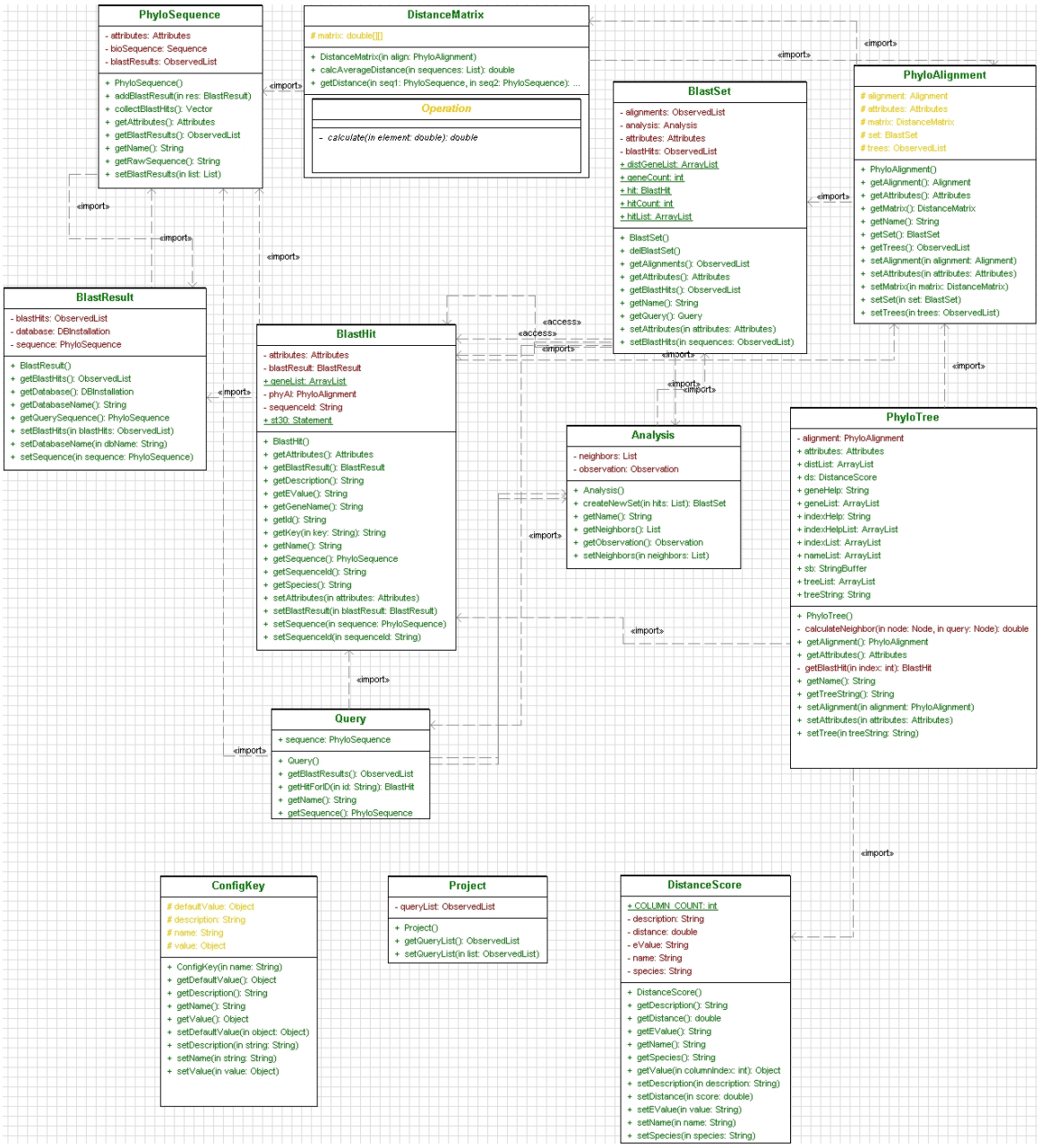


Abbildung 4.3: Auszug aus dem Klassendiagramm des Datenmodells

PhyloSequence

Die Klasse „PhyloSequence“ stellt eine Gensequenz dar, die entweder als DNA- oder als Proteinsequenz vorliegen kann.

Query

In der Klasse „Query“ ist der ORF hinterlegt, welcher vom System analysiert werden soll. Die Gensequenz des jeweiligen Queries enthält die „PhyloSequence“ mit der der betreffende Query verbunden ist.

Project

„Project“ fasst alle zur Analyse bestimmten Queries in einer Liste zusammen um eine nacheinander abfolgende Verarbeitung der Queries während einer Arbeitssitzung von PhyloGena zu gewährleisten. Die Klasse „Project“ dient hierbei als eine Organisationseinheit.

ConfigKey

Die Konfigurationsparameter, die zur Durchführung eines Blastprozesses benötigt werden, werden in der Klasse „ConfigKey“ gespeichert.

BlastResult

„BlastResult“ repräsentiert das Resultat einer Blastabfrage. Es ist mit dem betreffenden Query verknüpft und beinhaltet sämtliche BlastHit-Objekte.

BlastHit

Jeder einzelne Treffer während eines Blastprozesses bildet einen „BlastHit“. Die Gensequenz des lokalisierten Treffers wird in einer separaten „PhyloSequence“ gespeichert. Alle weiteren Daten, die während der Blast-abfrage gewonnen werden, wie z. B. die Spezies, werden in den jeweiligen Attributen dieser Klasse hinterlegt.

BlastSet

„BlastSet“ beinhaltet eine Anzahl von „BlastHit“-Objekten, die für die phylogenetische Analyse ausgewählt wurden. Der „BlastSet“ dient als Grundlage zur Berechnung eines „PhyloAlignment“.

PhyloAlignment

Das „PhyloAlignment“ repräsentiert ein Alignment, das von einem externen Alignmentprogramm berechnet wurde. Zur Berechnung des Alignments wird der „BlastSet“ und der „Query“ verwendet. Mit Hilfe des „PhyloAlignment“ können phylogenetische Bäume berechnet werden.

PhyloTree

„PhyloTree“ stellt einen auf Basis des „PhyloAlignment“ berechneten phylogenetischen Baum dar. Dieser lässt sich mit externen Visualisierungsprogrammen präsentieren.

DistanceMatrix

Mit Hilfe von „DistanceMatrix“ werden die Abstände der Verwandschaftsgrade der einzelnen Gensequenzen berechnet.

DistanceScore

„DistanceScore“ beinhaltet die statistischen Daten der auf Grundlage der „DistanceMatrix“ berechneten Daten.

Analysis

„Analysis“ repräsentiert alle evolutionären Nachbarn zu „Query“.

4.5 Konzeptioneller Aufbau des Abfrage- und Visualisierungsmodells von PhyloGena

Abbildung 4.4 stellt die Klassen des Abfragemodells von PhyloGena dar.

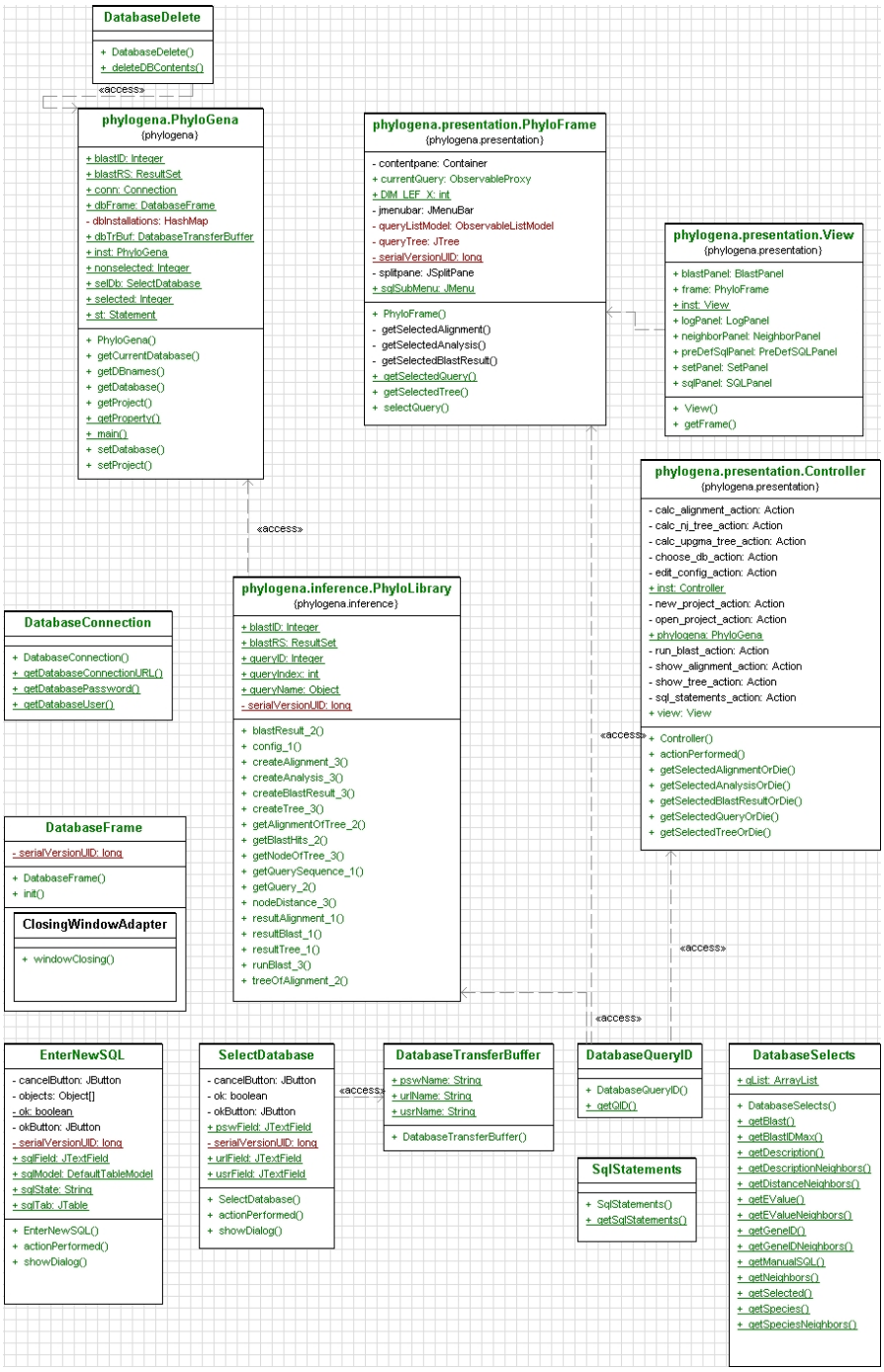


Abbildung 4.4: Auszug aus dem Klassendiagramm des Abfrage- und Visualisierungsmodells

PhyloGena
Diese Klasse beinhaltet die main()-Funktion und stellt somit den Start des Programms dar. Der Verbindungsaufbau zur Datenbank findet ebenfalls in dieser Klasse statt.

PhyloFrame
Die Klasse „PhyloFrame“ repräsentiert die GUI (Graphic User Interface) von PhyloGena.

DatabaseDelete
„DatabaseDelete“ hinterlegt die Befehle zum Löschen der Datenbankeinträge.

View
In der Klasse „View“ vereinen sich die Operationen zur Visualisierung der einzelnen Panels.

Controller
„Controller“ präsentiert sich als Action-Handler. Es werden unterschiedliche Operationen ausgeführt, die eine Interaktion mit dem Benutzer erfordern.

PhyloLibrary
„PhyloLibrary“ bietet eine Sammlung von Funktionen, welche mit Prologregeln interagieren.

DatabaseConnection
Diese Klasse liefert relevante Informationen zum Aufbau einer Verbindung zur Datenbank (Passwort, etc.)

DatabaseFrame
„DatabaseFrame“ initialisiert eine GUI für diverse Inter-aktionen mit dem Nutzer.

EnterNewSQL

Die Klasse „EnterNewSQL“ startet einen Dialog mit dem Nutzer, führt Operationen aus und liefert Ergebnisdaten zur Visualisierung.

SelectDatabase

„SelectDatabase“ initialisiert einen Dialog in Bezug auf die Datenbankverbindung.

DatabaseTransferBuffer

Die Objekte dieser Klasse dienen als Zwischenspeicher der „SelectDatabase“-Eingabe.

DatabaseQueryID

„DatabaseQueryID“ berechnet die zur Visualisierung der Blastergebnisse hinsichtlich eines bestimmten Queries korrekte blastID.

SqlStatements

Diese Klasse beinhaltet die Informationen zur Präsentation der vordefinierten SQL-Abfragen.

DatabaseSelects

Die Klasse „DatabaseSelects“ liefert sämtliche SQL-Abfragen, die für eine Standardprozedur von PhyloGena nötig sind.

5 Realisierung

5.1 Datenbankimplementierung

Um alle nötigen Informationen, die zum einen von PhyloGena zur Weiterverarbeitung benötigt werden und zum anderen dem Biologen bei nachträglicher Analyse der Daten ein möglichst genaues Bild der Homologie zu dem jeweiligen Query zu ermöglichen, zu sammeln, sind in der Datenbank neun Tabellen implementiert. Im Folgenden werden die verwendeten Datentypen sowie die Datenbankverbindung näher erläutert.

5.1.1 Datentypen

Für die Spaltengenerierung finden drei Datentypen Anwendung:

- VARCHAR(n),
- FLOAT(n,m) und
- INTEGER(n).

Die Felder, deren Attribute mit VARCHAR(n) deklariert sind, können beliebige alphanumerische Zeichen sowie Sonderzeichen enthalten. Die maximale Länge für den Datenwert beträgt n Byte und ist auf 4000 Byte begrenzt. VARCHAR als Datentyp ist variabel, das bedeutet, dass nur die tatsächlich eingegebene Zeichenkette abgespeichert wird. Bei Zeichenketten, die kürzer als n sind, werden so im Gegensatz zu dem Datentyp CHAR(n) keine Leerzeichen ergänzt.

Mit FLOAT(n,m) initialisierte Felder spezifizieren Gleitkommazahlen. Hierbei dient n zur Bestimmung der Gesamtziffernzahl und m repräsentiert die Anzahl der Nachkommastellen. Bei dem Datentyp FLOAT(n,m) besteht durch die Negativangabe von m die Möglichkeit, die letzte visualisierte Nachkommastelle zu runden.

Mit dem Datentyp INTEGER(n) können Ganzzahlen mit einer Größe von bis zu 4 Byte in die jeweilige Tabelle abgelegt werden. Dies entspricht einem vorzeichenunbehafteten Maximalwert von 4.294.967.295.

5.1.2 Datenbankverbindung

Die Datenbank ist auf einem MySQL-Datenbankserver implementiert. Somit wird erreicht, dass alle Nutzer auf die Daten bereits durchgeführter Blastläufe zugreifen können. Um von PhyloGena auf die Datenbank zugreifen zu können, kommt der bereits erwähnte MySQL Connector/J zum Einsatz. Direkt beim Start von PhyloGena wird ein Verbindungsfenster geöffnet, in das die nötigen Informationen zur Verbindung eingetragen werden.

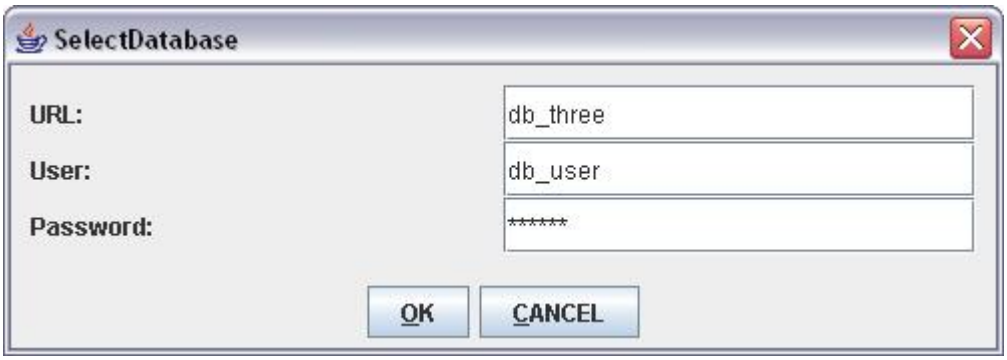


Abbildung 5.1: Verbindungsfenster PhyloGena - Datenbank

Die Verbindung zur Datenbank erfolgt unmittelbar danach und bleibt solange bestehen bis die Sitzung mit PhyloGena beendet wird. Zur Kontrolle der Daten sollte es administrativ untersagt sein, mehrere Sitzungen gleichzeitig zu erlauben. Somit kann sichergestellt werden, dass der Nutzer nicht auf falsche Datensätze zugreifen kann. Wenn zwei oder mehr Sitzungen parallel laufen, kann es zu Überschneidungen bei den Primärschlüsseln kommen, was zu Fehlermeldungen in der Datenbank führen kann. Falls keine Doppelnennungen der primären Schlüssel auftreten, wäre es weiterhin möglich, dass bei der Visualisierung der Daten auf falsche Datenwerte zugegriffen wird, da die abgefragte „blastID“ nicht mehr mit der tatsächlichen „blastID“ übereinstimmt.

5.2 Datenakquise

Die Sammlung der benötigten Daten stellte die größte Herausforderung der hier vorliegenden Diplomarbeit dar. In erster Linie musste festgelegt werden, welche Daten wie abgelegt werden. Des Weiteren mussten die Daten im Ablauf von PhyloGena lokalisiert und erfasst werden. In den folgenden Unterkapiteln wird auf die Datensammlung der einzelnen Tabellen in der Datenbank eingegangen.

5.2.1 Tabelle „config“

Die Datenwerte der Attribute „evalueCutoff“, „alignMethod“, „blastRule“, „selectionRule“, „treeMethod“, „maxSequences“ und „analysisName“ sind Startwerte, welche vor der Blastabfrage festgelegt werden und die Suche nach Homologen beeinflussen. Sie werden aus dem Konfigurationsfenster als String bzw. Integer ausgelesen und in eine entsprechende Liste eingetragen.

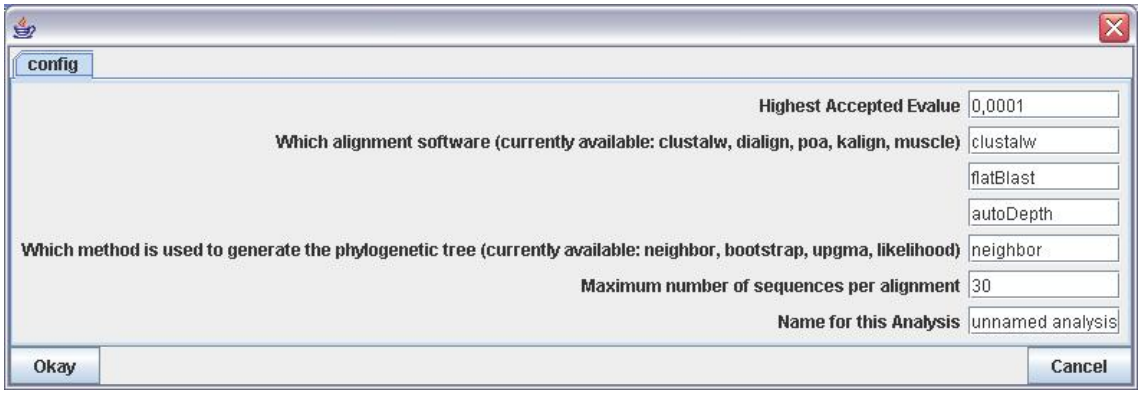


Abbildung 5.2: Konfigurationsfenster PhyloGena

Diese Liste wird mit Hilfe einer for-Schleife in der Klasse „PhyloLibrary“ ausgelesen und mittels des SQL-Befehls INSERT INTO in die Tabelle der Datenbank geschrieben.

Der jeweilige Wert für „blastID“ wird unmittelbar nach dem Starten von PhyloGena ebenfalls in der Klasse „PhyloLibrary“ berechnet, indem der höchste Wert der „blastID“ in der bestehenden Tabelle abgefragt und anschließend inkrementiert wird. Bei erstmaliger Beanspruchung der Datenbank ist der Wert 0. Daraus ergibt sich, dass der erste durchgeführte Blastprozess mit 1 initialisiert wird.

5.2.2 Tabelle „query“

Das Attribut „querySeq“ der Tabelle „query“ speichert die Gensequenz des Queries, welche aus der Klasse „PhyloSequence“ ausgelesen wird. Dies gewährleistet, durch gezielte SELECT-Abfragen, bestimmte Gensequenzen aus der Datenbank zu lokalisieren und diese in ein neues Projekt von PhyloGena einzufügen. Mit einem solchen Projekt ist ein erneuter Blastprozess und eine neue, damit verbundene, Analyse möglich.

5.2.3 Tabelle „blast“

Die Tabelle „blast“ beinhaltet alle Datenwerte, die während eines Blastlaufs gesammelt werden. Die Klasse „BlastHit“ verfügt hierzu über eine Reihe von Attributen, welche die jeweilige Information bezüglich der lokalisierten homologen Gensequenz speichern. Jedes Attribut setzt sich aus einem Namen und dem dazugehörigen Wert zusammen. Dies ermöglicht unter Berücksichtigung des gewünschten Namens das Auslesen der betreffenden Informationen. Nachdem alle Informationen einer Gensequenz zwischengespeichert wurden, werden sie in die entsprechenden Spalten der Tabelle blast abgelegt. Wenn durch den Blastprozess ein neuer Treffer gefunden wird, werden die Attribute der Klasse „BlastHit“ mit den aktuellen Informationen überschrieben, und der Übertragungsvorgang an die Datenbank beginnt von neuem. Auf diese Weise werden nach und nach alle Informationen der in Betracht kommenden homologen Gensequenzen in die Datenbank abgelegt.

5.2.4 Tabelle „blast_result“

Wie in dem Kapitel 3.1.6 bereits erwähnt findet bei PhyloGena eine Vorauswahl der Sequenzen statt, um mit den Wahrscheinlichsten das Alignment zu bestimmen und abschließend den phylogenetischen Baum zu berechnen. Diese Vorauswahl findet in der Tabelle „blast_result“ Berücksichtigung. Hierzu werden die Sequenzen, welche die Vorauswahl bilden, in eine separate Liste geschrieben. Die Einträge dieser Liste werden mit den Einträgen einer anderen Liste verglichen, in der alle Gensequenzen eingetragen sind. Bei Übereinstimmung der jeweiligen Sequenz wird eine 1 in der Tabelle generiert, die übrigen Einträge werden mit einer 0 versehen und signalisieren so, dass sie nicht zur Bildung des Alignment beitragen. Sowohl das Regelwerk zur Bestimmung der Vorauswahl als auch die Übertragung der Datenwerte in die dafür

vorgesehene Tabelle werden in der Klasse „BlastSet“ ausgeführt. Zwar bleiben die Daten zur weiterführenden Berechnung weiterhin in den Attributen der Klasse bestehen, jedoch können Referenzen, welche auf die Listen verweisen, direkt nach der Übertragung aufgehoben werden.

5.2.5 Tabelle „neighbor“

Die Tabelle „neighbor“ dient maßgeblich der Speicherung der Distanzen einzelner Gensequenzen zum gesuchten Query. Die Distanz stellt die Entfernung und somit einen evolutionären Ablauf zwischen den Sequenzen in einer Baumstruktur dar. Zur Berechnung dient die Funktion calculateNeighbor() in der Klasse „PhyloTree“. Die Ergebnisse werden in Attribute geschrieben, welche neben der Angabe der Distanz auch den Namen der Sequenz und die Spezies des Gens enthalten. Mit Hilfe dieser Attribute werden die Informationen für die Tabelle ausgelesen und abgelegt.

5.2.6 Tabelle „minNeighbor“

Einen schnellen Überblick des gesuchten Queries erhält man mit der Tabelle „minNeighbor“. Sie stellt eine Spezialisierung der Tabelle „neighbor“ dar. Hier wird lediglich die Gensequenz mit ihrer Distanz abgelegt, die die geringste evolutionäre Entfernung zum Query aufweist. Diese Eigenschaft kann bei späteren SELECT-Abfragen eminent wichtig sein. Da SELECT als Unterabfrage genau einen Rückgabewert erwartet, vermeidet diese Verfeinerung der Tabelle „neighbor“ eventuelle Mehrdeutigkeiten in Unterabfragen.

5.2.7 Tabelle „sequence“

Wie bereits erwähnt fungiert die Tabelle „sequence“ als Verbindung zwischen dem eigentlichen Namen und dem entsprechenden numerischen Wert einer Gensequenz, der während einer Berechnung mit PhyloGena genutzt wird. Diese Zuweisung ist im Hinblick auf das Alignment unumgänglich, da die eigentliche Darstellung des Alignments erst durch das externe Visualisierungsprogramm JalView entsteht. Die Tabelle „sequence“ stellt somit eine Hilfstabelle für das Alignment dar. Im Verlauf der Berechnung wird die Gensequenz wie folgt benannnt:

sequence + [Index der jeweiligen Sequenz aus der Liste hitList].

Bei der Bestimmung des Alignment entsteht dadurch eine unbestimmte Abfolge der Indize. Um die Sequenznummern ihrem ursprünglichen Namen wieder zuordnen zu können, wird eine Liste angelegt, welche nur die Indexwerte als Integer speichert. Anschließend werden die Namen anhand dieser Indexliste wieder den Sequenzen zugewiesen und in der Klasse „PhylipAlignmentWriter“ an die Datenbank übertragen.

5.2.8 Tabelle „alignment“

Die Tabelle „alignment“ besteht aus vier Spalten. Zwei der Spalten bilden die „blastID“ und die „blastVersion“, welche in allen Tabellen zu finden sind, die restlichen beiden Spalten definieren die Zuweisung der Gensequenzen zu den entsprechenden Alignments. Bei den Alignments werden auch die möglicherweise vorliegenden Lücken erfasst und als Bindestrich dargestellt. Hintergrund hierbei ist, dass sich bei späterem Bedarf eine Visualisierung des Alignments mit externen Programmen wie z. B. JalView [39] realisieren lässt. Die Liste, welche die Symbole für das Alignment speichert, ist ebenfalls in der Klasse „PhylipAlignmentWriter“ realisiert. Die Symbolik des Alignments ist ein Resultat des durchgeführten Blastlaufs und wird ebenfalls in Attributen der Klasse hinterlegt.

5.2.9 Tabelle „tree“

Programme, die wie beispielsweise ATV zur Visualisierung von phylogenetischen Bäumen konzipiert sind, lesen die Daten zur Darstellung des Baumes in dem New Hampshire Format ein. Das New Hampshire Format ist eine funktionsähnliche Beschreibung des phylogenetischen Baums. Zur Veranschaulichung soll ein einfaches Beispiel dienen.

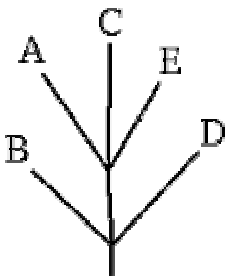


Abbildung 5.3: Beispiel eines gewurzelten Baumes;
Quelle: [40]

Der, in Abbildung 5.3, aufgeführte Baum wird im New Hampshire Format wie folgt beschrieben:

(B:6.0,(A:5.0,C:3.0,E:4.0):5.0,D:11.0);

Hierbei repräsentieren die Buchstaben das jeweilige Gen im Baum und die Zahlen stellen die Distanz zu dem am nächsten liegenden Knoten dar. Durch die Klammerung wird die Zugehörigkeit zu einem Knoten des Baums verdeutlicht.

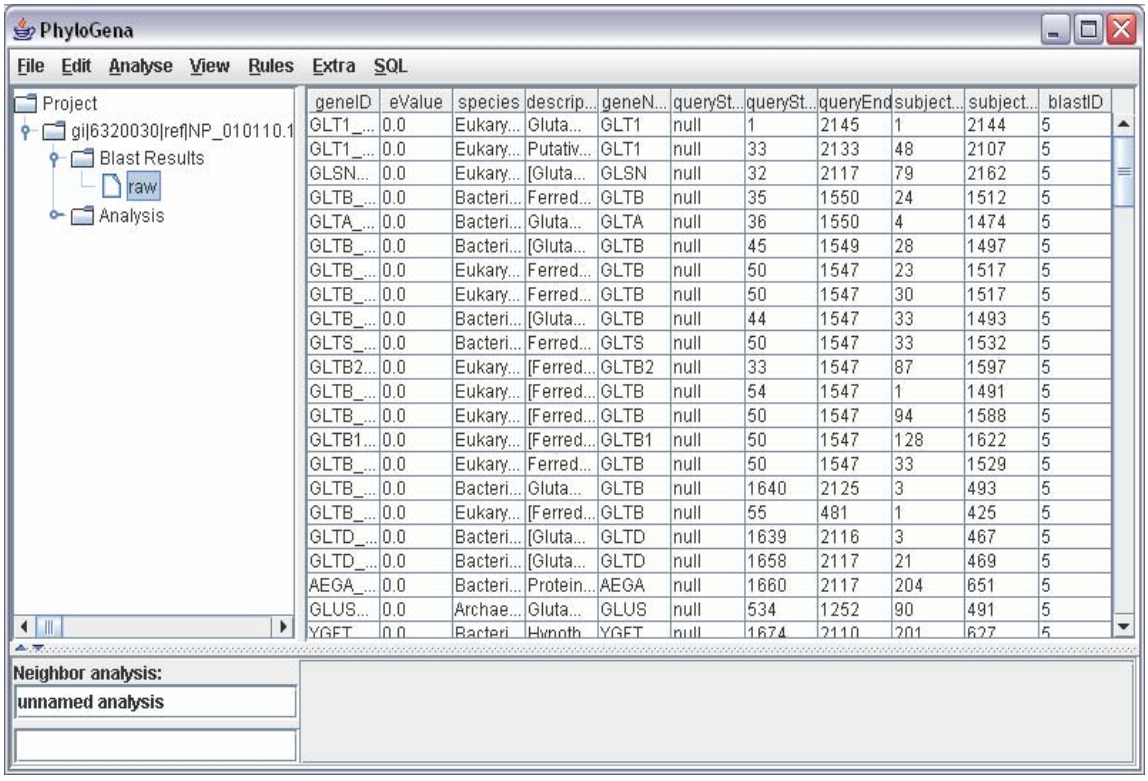
Die Bestimmung des New Hampshire Formates wird in der Klasse „PhyloTree“ eingeleitet und durch das externe Programm ATV berechnet. Der so entstanden Code wird ebenfalls in der Klasse „PhyloTree“ als String in die Spalte „nhCode“ der Tabelle „tree“ geschrieben. Zuvor werden jedoch, wie in Kapitel 5.2.7 beschrieben, die Namen der einzelnen Sequenzen in dem vorliegenden Code ersetzt. Die so entstandene Zuweisung zwischen „blastID“ und „nhCode“ ermöglicht die direkte Darstellung eines Baumes aus der Datenbank heraus.

5.3 Visualisierung der Daten in PhyloGena

Nachdem die Suche nach homologen Gensequenzen zum gesuchten Query abgeschlossen ist und alle Datenwerte in der Datenbank abgelegt sind, lassen sich diese zur Analyse wieder in PhyloGena darstellen. Um diese Visualisierung zu gewährleisten, kommen verschiedene SELECT-Abfragen zum Einsatz. Weiterhin ist ein geeignetes Panel notwenig, um die gelieferten Daten übersichtlich darzustellen. Im Folgenden werden diese Punkte näher erläutert.

5.3.1 Panels

Die zur Repräsentation der Daten nötigen Panels enthalten alle ein Modell, welches es ermöglicht, die Datenwerte ähnlich der Datenbank in Tabellen anzuzeigen.



geneID	eValue	species	descrip...	geneN...	querySt...	querySt...	queryEnd	subject...	subject...	blastID
GLT1_...	0.0	Eukary...	Gluta...	GLT1	null	1	2145	1	2144	5
GLT1_...	0.0	Eukary...	Putativ...	GLT1	null	33	2133	48	2107	5
GLSN...	0.0	Eukary...	[Gluta...	GLSN	null	32	2117	79	2162	5
GLTB_...	0.0	Bacteri...	Ferred...	GLTB	null	35	1550	24	1512	5
GLTA_...	0.0	Bacteri...	Gluta...	GLTA	null	36	1550	4	1474	5
GLTB_...	0.0	Bacteri...	[Gluta...	GLTB	null	45	1549	28	1497	5
GLTB_...	0.0	Eukary...	Ferred...	GLTB	null	50	1547	23	1517	5
GLTB_...	0.0	Eukary...	Ferred...	GLTB	null	50	1547	30	1517	5
GLTB_...	0.0	Bacteri...	[Gluta...	GLTB	null	44	1547	33	1493	5
GLTS_...	0.0	Bacteri...	Ferred...	GLTS	null	50	1547	33	1532	5
GLTB2...	0.0	Eukary...	[Ferred...	GLTB2	null	33	1547	87	1597	5
GLTB_...	0.0	Eukary...	[Ferred...	GLTB	null	54	1547	1	1491	5
GLTB_...	0.0	Eukary...	[Ferred...	GLTB	null	50	1547	94	1588	5
GLTB1...	0.0	Eukary...	[Ferred...	GLTB1	null	50	1547	128	1622	5
GLTB_...	0.0	Eukary...	Ferred...	GLTB	null	50	1547	33	1529	5
GLTB_...	0.0	Bacteri...	Gluta...	GLTB	null	1640	2125	3	493	5
GLTB_...	0.0	Eukary...	[Ferred...	GLTB	null	55	481	1	425	5
GLTD_...	0.0	Bacteri...	[Gluta...	GLTD	null	1639	2116	3	467	5
GLTD_...	0.0	Bacteri...	[Gluta...	GLTD	null	1658	2117	21	469	5
AEGA_...	0.0	Bacteri...	Protein...	AEGA	null	1660	2117	204	651	5
GLUS...	0.0	Archae...	Gluta...	GLUS	null	534	1252	90	491	5
YGET...	0.0	Bacteri...	Hypoth...	YGET...	null	1674	2110	201	627	5

Abbildung 5.4: Visualisierung der Daten aus der Tabelle „blast“ zu einer Blastsuche.

Während der Abfrage an die Datenbank wird die Anzahl der Einträge ermittelt und die nötige Zeilenanzahl der darzustellenden Tabelle festgelegt. Das nun die

Tabelle darstellende Panel wird aus Gründen der Übersichtlichkeit an die dafür vorgesehen Position in dem Ursprungsframe von PhyloGena eingefügt.

5.3.2 SELECT-Abfragen

Grundlage der SELECT-Abfragen ist die jeweilige „blastID“. In der Datenbank werden alle getätigten Blastergebnisse gehalten, wogegen bei der Visualisierung in PhyloGena jedoch nur das aktuelle Ergebnis von Interesse ist. Das bedeutet, dass bei dem SELECT-Befehl eine Einschränkung hinsichtlich der „blastID“ getroffen werden muss.

Falls, wie in Abbildung 5.4 gezeigt, lediglich ein Query in PhyloGena geladen wird, ist hierfür nur die „blastID“ interessant, welche den höchsten Wert einnimmt. Werden jedoch mehrere Queries in PhyloGena geladen, die nacheinander während einer Sitzung abgearbeitet werden sollen, ist eine andere Vorgehensweise notwendig.

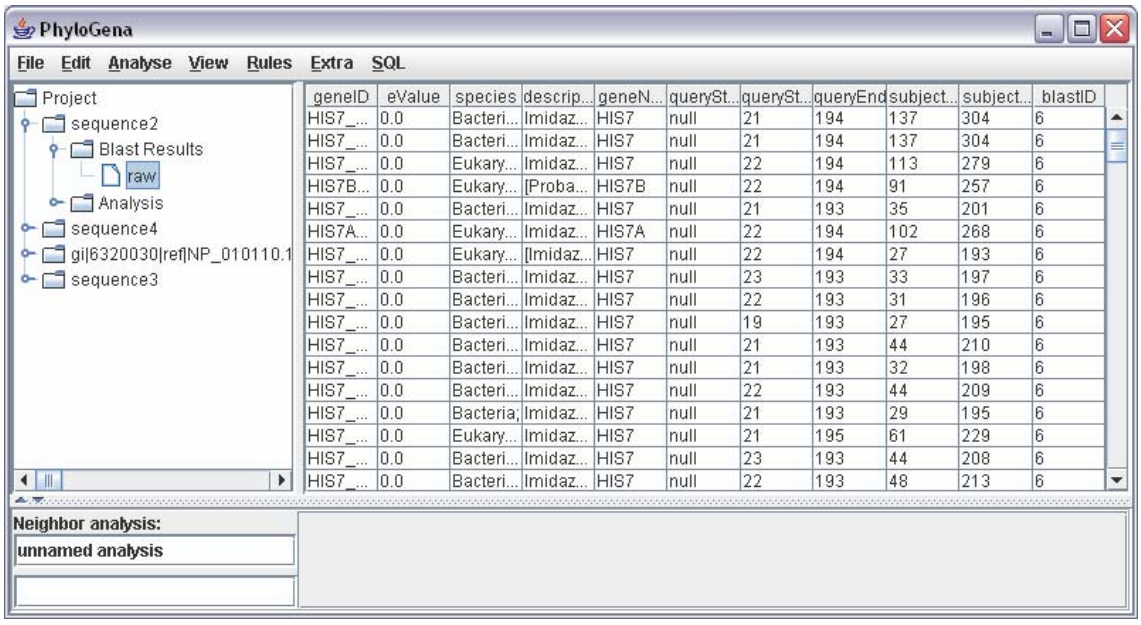


Abbildung 5.5: Visualisierung der Daten aus der Tabelle „blast“ zu einer Blastsuche bei mehreren Queries.

Hierbei werden die verschiedenen Queries in umgekehrter Reihenfolge ihres Ladens in eine Liste geschrieben, sodass, wie in Abb. 5.5 gezeigt, die sequence3 den Index 0 besitzt. Mit Hilfe dieser Liste lässt sich die Zuweisung der „blastID“ realisieren. Zunächst wird die „blastID“ mit dem höchsten Wert aus

der Datenbank abgefragt und anschließend der Wert des Index von der „blastID“ subtrahiert. Der dadurch gewonnene Wert entspricht der „blastID“ des jeweiligen Queries.

Alle SELECT-Abfragen, die zur Visualisierung der einzelnen Queries relevant sind, sind in der Klasse „DatabaseSelects“ vereint. Sie werden in den einzelnen Funktionen aufgerufen, die durch Selektion der Symbole in der Baumhierarchie gestartet werden. Dadurch werden die Daten erst dann abgerufen, wenn sie tatsächlich benötigt werden.

Die Ergebnisse einer SELECT-Abfrage werden in einem „ResultSet“, welches durch die Bibliothek java.sql bereitgestellt wird, als Objekte gespeichert. Durch Einlesen der Objekte in ein Tabellenmodell können die Daten dem jeweiligen Panel übergeben und dadurch in PhyloGena präsentiert werden.

5.4 Vordefinierte SQL-Befehle

Um die Vorteile einer Datenbank hinsichtlich einer übersichtlichen Datenerfassung zu nutzen, wurde in die grafische Benutzeroberfläche von PhyloGena ein Untermenü implementiert, welches vordefinierte SELECT-Abfragen beinhaltet. Dieser Baustein ist als Basis realisiert und einfach um weitere Abfragen zu ergänzen. Im Verlauf der Anwendung der Software wird sich zeigen, welche weiteren SELECT-Befehle in diese vordefinierte Liste zu ergänzen sind.

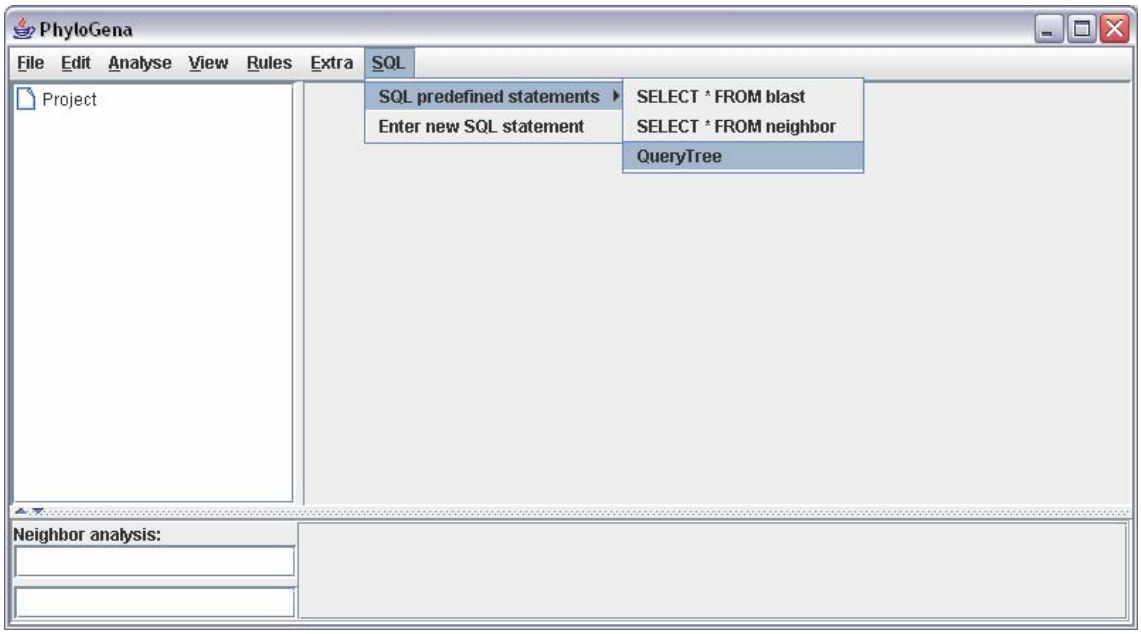


Abbildung 5.6: Untermenü "SQL predefined statements" zur Durchführung von SELECT-Abfragen

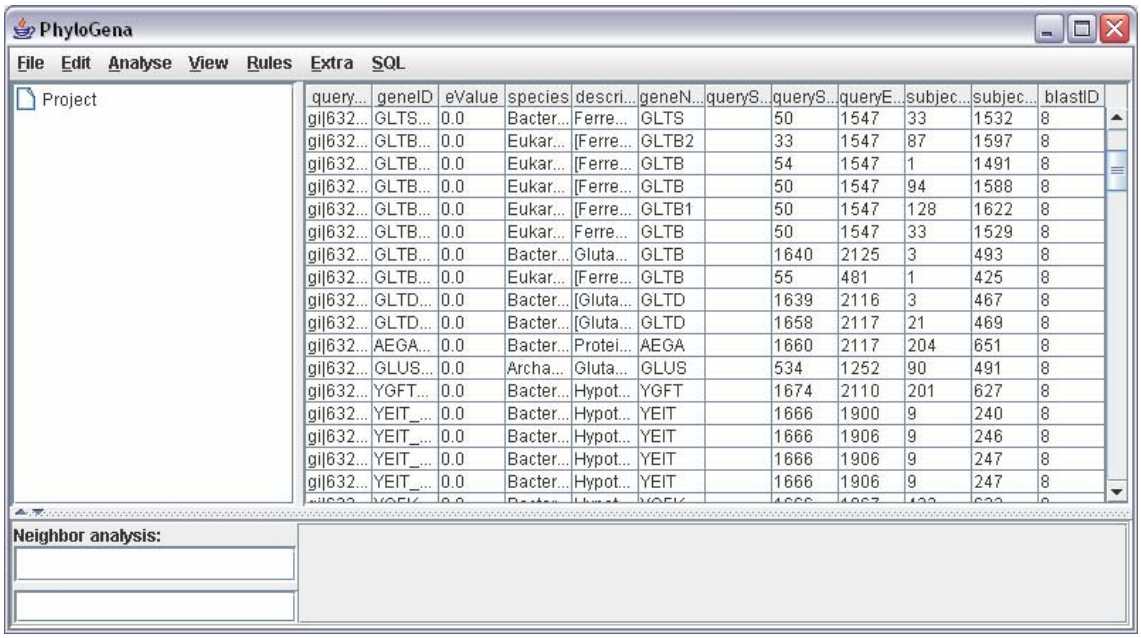
Um diese Liste mit weiteren Einträgen zu versehen, müssen Ergänzung in den Klassen „SqlStatements“ und „SubMenuListener“ vorgenommen werden. Die Klasse „SqlStatements“ dient lediglich zur Darstellung der Einträge, hier kann für den SELECT-Befehl ein beliebiger Name vergeben werden. Die eigentliche Bearbeitung des Befehls findet in der Klasse „SubMenuListener“ statt. Es wird ein ActionListener generiert, der die jeweilige Auswahl registriert und eine entsprechende Handlung ausführt. Obwohl diese Handlung unterschiedlichste Funktionen bergen kann, ist der Normalfall so ausgelegt, dass eine Tabelle generiert und ähnlich der Abbildung 5.4 in PhyloGena dargestellt wird, um dem Nutzer eine bessere Analyse der Daten zu ermöglichen.

5.5 Eingabe eigener SQL-Befehle

Der Menüeintrag SQL bietet neben den vordefinierten SQL-Befehlen auch die Möglichkeit eigene SELECT-Abfragen zu formulieren und diese auf die Datenbank anzuwenden. Da die Eingabe in SQL-Syntax erfolgen muss, wird ein Basiswissen an SELECT- Formulierung vorausgesetzt.

Abbildung 5.7: Eingabefenster für einen SQL-Befehl bei Aufruf des Menüeintrags "Enter new SQL statement"

Das Ergebnis wird ebenfalls ähnlich der Abbildung 5.4 in PhyloGena dargestellt.



query...	geneID	eValue	species	descri...	geneN...	queryS...	queryS...	queryE...	subjec...	subjec...	blastID
gi|632...	GLTS...	0.0	Bacter...	Ferre...	GLTS		50	1547	33	1532	8
gi|632...	GLTB...	0.0	Eukar...	[Ferre...	GLTB2		33	1547	87	1597	8
gi|632...	GLTB...	0.0	Eukar...	[Ferre...	GLTB		54	1547	1	1491	8
gi|632...	GLTB...	0.0	Eukar...	[Ferre...	GLTB		50	1547	94	1588	8
gi|632...	GLTB...	0.0	Eukar...	[Ferre...	GLTB1		50	1547	128	1622	8
gi|632...	GLTB...	0.0	Eukar...	Ferre...	GLTB		50	1547	33	1529	8
gi|632...	GLTB...	0.0	Bacter...	Gluta...	GLTB		1640	2125	3	493	8
gi|632...	GLTB...	0.0	Eukar...	[Ferre...	GLTB		55	481	1	425	8
gi|632...	GLTD...	0.0	Bacter...	[Gluta...	GLTD		1639	2116	3	467	8
gi|632...	GLTD...	0.0	Bacter...	[Gluta...	GLTD		1658	2117	21	469	8
gi|632...	AEGA...	0.0	Bacter...	Protei...	AEGA		1660	2117	204	651	8
gi|632...	GLUS...	0.0	Archa...	Gluta...	GLUS		534	1252	90	491	8
gi|632...	YGFT...	0.0	Bacter...	Hypot...	YGFT		1674	2110	201	627	8
gi|632...	YEIT_...	0.0	Bacter...	Hypot...	YEIT		1666	1900	9	240	8
gi|632...	YEIT_...	0.0	Bacter...	Hypot...	YEIT		1666	1906	9	246	8
gi|632...	YEIT_...	0.0	Bacter...	Hypot...	YEIT		1666	1906	9	247	8
gi|632...	YEIT_...	0.0	Bacter...	Hypot...	YEIT		1666	1906	9	247	8

Abbildung 5.8: Ergebnisanzeige nach Eingabe einer eigenen SELECT-Abfrage

Die Realisierung des Panels und der Behandlung des Abfrageergebnisses wurde in der Klasse „EnterNewSQL“ durchgeführt.

6 Zusammenfassung und Ausblick

6.1 Zusammenfassung

Im Rahmen dieser Diplomarbeit wurde eine Datenbankanbindung an das bereits bestehende Programm PhyloGena implementiert. Die Notwendigkeit für eine Datenbank kristallisieren sich durch die folgenden beiden Aspekte heraus:

1. Um eine phylogenetische Analyse hinsichtlich eines unbekannten Queries zu ermitteln, führt das Programm PhyloGena rechenintensive Operationen durch. Mit Hilfe einer Datenbank, welche die Informationen einer solchen phylogenetischen Analyse speichert, wird gewährleistet, dass man jederzeit auf die bereits ermittelten Daten zugreifen kann ohne eine erneute Blastsuche zu initiieren. Dieser Vorteil berücksichtigt auch Vergleiche verschiedener Queries miteinander, sofern zu den Queries bereits eine separate phylogenetische Analyse vorliegt.
2. Maßgeblich zur Visualisierung der berechneten und ermittelten Daten hält PhyloGena sämtliche gewonnen Informationen im Arbeitsspeicher. Dieser Umstand führt dazu, dass die möglichen Analysen einer Session durch die Größe des Arbeitsspeichers begrenzt sind. Durch die Anbindung einer Datenbank und die Übertragung der Daten während der Laufzeit von PhyloGena ist diese Begrenzung aufgehoben. Die Visualisierung der phylogenetischen Analyse erfolgt mit den Daten aus der Datenbank zu dem jeweils erforderlichen Zeitpunkt.

Hinsichtlich der Attribute und Datenwerte, welche die Relationen der Datenbank bilden, wurden die Hintergründe der phylogenetischen Analyse und die Arbeitsweise von PhyloGena beschrieben. Außerdem wurden der Aufbau der Datenbank, die Datenakquise und die Abfragen der jeweilgen Tabelleninhalte erläutert.

Bei der Weiterentwicklung von PhyloGena und der Visualisierung der Daten wurde darauf geachtet, das einheitliche Bild der benutzerfreundlichen grafischen Oberfläche beizubehalten. Auch Abfragen, welche direkt aus der Benutzeroberfläche von PhyloGena abgesetzt werden können und somit ein breites Spektrum für die Datenverarbeitung liefern, fügen sich nahtlos in das bestehende Bild von PhyloGena ein und runden so die Datenbankimplemen-

tierung ab. Hierbei lag das Hauptaugenmerk auf Flexibilität, sodass eine einfache Erweiterung des Abfragesystems jederzeit gewährleistet ist.

6.2 Ausblick

Die zeitaufwändigste Operation einer phylogenetischen Analyse stellt der Blastprozess dar. Durch die im Laufe dieser Diplomarbeit entworfene Datenbank besteht die Möglichkeit, diese Operation zeitlich zu optimieren, indem Daten zu Gensequenzen, welche sich bereits in der Datenbank befinden, bei der Analyse eines neuen Queries nicht erneut von Sekundärdatenbanken abgefragt werden, sondern direkt aus den bestehenden Datenbankeinträgen abgefragt werden. Besonders bei häufig wiederkehrenden Gensequenzen würde dies eine enorme Zeitersparnis darstellen. Um diese zu gewährleisten, sind allerdings noch weiterführende Arbeiten von Nöten.

Zuerst sollten neue Abfrageregeln in PhyloGena definiert werden, die bedingen, dass im Anschluss an die Abfrage von Sequenzdatenbanken nach Homologen die lokale Datenbank nach bereits getätigten Einträgen durchsucht und nur diejenigen, welche nicht lokal auftauchen, extern abgefragt werden. Damit diese Daten aus der Struktur der Datenbank abgefragt und dem neuen Blastergebnis zugeordnet werden können, müssten sie kopiert und mit einer entsprechenden „blastID“ versehen werden. Die größte Problematik besteht hierbei in der hierarchischen Struktur von PhyloGena, welche, bedingt durch die formulierten Regeln zur Erstellung einer phylogenetischen Analyse, vorsieht, dass eine schrittweise Abarbeitung der durch Blast gefundenen Homologen erfolgt. Zur Realisierung dieser Datenbankabfrage müssen demzufolge neue Prologregeln erstellt und bestehende Regeln optimiert werden.

Diese Problematik bezieht sich ebenfalls auf Blastprozesse zu bereits getätigten Analysen. Für einen Biologen kann es interessant sein, zu einem bestimmten Query Analysen mit unterschiedlichen Konfigurationsparametern durchzuführen. Hierbei sollte darauf verzichten werden, bei jeder anstehenden Analyse eine Blastabfrage zu starten. Die zur Durchführung der einzelnen Analysen benötigten Daten werden durch die Datenbank bereitgestellt und bieten somit die Grundlage zur Bestimmung und Berechnung der jeweiligen Datenwerte bezüglich unterschiedlicher Konfigurationsparametern. Das Speichern der neuen Daten wird mit dem Attribut „blastVersion“ ermöglicht. Bei solch einer erneut durchgeführten Analyse würde die „blastID“ bestehen

bleiben, nur die „blastVersion“ sollte inkrementiert werden. Die Schwierigkeit hierbei liegt ebenfalls an dem hierarchischen Aufbau von PhyloGena. Um eine Datenbankabfrage anstatt eines Blastprozesses zu ermöglichen sind weitere Eingriffe in die Struktur der vorhandenen Prologregeln nötig.

Phylogenetische Analysen rücken bei vielen Wissenschaftlern hinsichtlich der Einordnung einer unbekannten Gensequenz mehr und mehr in den Vordergrund. Aus diesem Grund wäre es sinnvoll eine institutsübergreifende Wissensbasis zu schaffen, welche als Datenbank implementiert, die Daten aus PhyloGena repräsentiert. Der Grundstein für ein solches Unternehmen wäre die Verbreitung von PhyloGena und das Einrichten einer allgemein zugänglichen Datenbank, wie z.B. GenDB. GenDB ist ein Annotationssystem für prokaryotische Genome. Es wurde entwickelt, um dem Biologen eine benutzerfreundliche Annotationsplattform für diverse Genomprojekte zu bieten [41]. Zurzeit entwickelt das Max-Planck-Institut für Marine Mikrobiologie in Bremen Erweiterungen für GenDB, um es auf dem Gebiet der Vergleichenden Genomik einsetzen zu können [42].

Die Integration der während dieser Arbeit entstandenen Datenbank wäre einfach zu realisieren, indem standardisierte Attribute Verwendung finden. Eine Schwierigkeit bildet jedoch die Administration, man müsste sicherstellen, dass bei höher frequentem Zugriff auf die Datenbank keine Daten verloren gehen und die Zuweisung der einzelnen Datenwerte der jeweiligen „blastID“ entspricht. Vorstellbar wäre in diesem Zusammenhang, dass beim Start von PhyloGena eine gesicherte Verbindung zur Datenbank aufgebaut wird, die den Zugriff auf alte Datenwerte und durch die jeweilige Session entstandene Daten gewährleistet. Daten, welche durch einen anderen Nutzer zur selben Zeit generiert werden, werden nicht berücksichtigt. Am Ende einer Session müssten die Tabelleneinträge aktualisiert werden, so dass keine Mehrdeutigkeit hinsichtlich der Zuweisung der „blastID“ auftreten kann.

Anhang A: CD-ROM

Die CD-ROM ist in zwei Gruppen gegliedert, welche sich jeweils in separaten Dateiordnern befinden:

Dokumente:

- Diplomarbeit_PGDB.pdf
- Fachkonzept.xml
- Tabellenentwurf.xml
- Präsentation_PGDB.pps (wird nach dem Kolloquium eingereicht)

Quellcode und Projekt:

- PhyloGena (Eclipse-Workspace[1])
  - phylogena
  - database
  - atv
  - jalview
  - 2p
- Executable Jar Files

[1] http://www.eclipse.org

Anhang B: Literaturverzeichnis

[1] Webpräsenz des Alfred-Wegener-Institutes für Polar- und Meeres-forschung
http://www.awi.de

[2] MÜLHARDT, Cornel: „Der Experimentator: Molekularbiologie/Genomics“. Dritte Auflage. Spektrum Akademischer Verlag Heidelberg · Berlin, 2002

[3] Webpräsenz von Access Excellence: DNA-Molekül
http://www.accessexcellence.org/RC/VL/GG/dna_molecule.html

[4] Webpräsenz des National Human Genome Research Institute: DNA
http://www.genome.gov/Pages/Hyperion/DIR/VIP/Glossary/Illustration/dna...

[5] Dr.-Ing. KOWARSCHIK, Markus: „At Home In The Universe“.
Lehrstuhl für Informatik, Universität Erlangen-Nürnberg, WS99/00
http://www10.informatik.uni-erlangen.de/~markus/ws99_00/AHitU/bastian/ahitu_c2.htm

[6] Online Enzyklopädie; Suchbegriff: „DNA-Sequenzierung“
http://de.wikipedia.org/wiki/DNA-Sequenzierung

[7] LENGAUER, Thomas: „Bioinformatics: From the Pre-genomic to the Post-genomic Era”. Journal: ERCIM News 43, 2000
http://www.ercim.org/publication/Ercim_News/enw43/lenggauer.html

[8] TECH, Maike: „Genannotation bei Prokaryoten“.
Institut für Mikrobiologie und Genetik, Universität Göttingen, WS05/06
http://www.gobics.de/lectures/ws05/Einfuehrung_bi_folien/07_genePredProka.pdf

[9] LEHNINGER, Albert L.: „Prinzipien der Biochemie“.
Erste Auflage. Walter de Gruyter, 1987

[10] Prof. SCHULTZ, Joerg: Skript zur Vorlesung „Bioinformatik II“.
Julius-Maximilians-Universität Würzburg, SS06
http://bioapps.biozentrum.uni-wuerzburg.de/~binf009/Vorlesung/2006_06_20_pre.pdf

[11] Online Enzyklopädie; Suchbegriff: „Paralogie“
http://de.wikipedia.org/wiki/Paralogie

[12] GALPERIN, Michael Y.: „The Molecular Biology Database Collection: 2007 update“. Journal: Nucleic Acids Research 35, 2007
http://nar.oxfordjournals.org/cgi/content/abstract/35/suppl_1/D3

[13] Online Enzyklopädie; Suchbegriff: „Sequenzdatenbank“
http://de.wikipedia.org/wiki/Sequenzdatenbank

[14] Webpräsenz von EBI: News
http://www.ebi.ac.uk/embl/News/news.html

[15] Webpräsenz von EBI: Statistiken
http://www3.ebi.ac.uk/Services/DBStats/

[16] Webpräsenz von Swiss-Prot
http://www.expasy.org/sprot/sprot_details.html

[17] Online Enzyklopädie; Suchbegriff: „Sequenzalignment“
http://de.wikipedia.org/wiki/Sequenzalignment

[18] MOUNT, David W.: „Bioinformatics: Sequence and Genome Analysis“. Erste Auflage. Cold Spring Harbor Laboratory Press, 2001

[19] JONES, Neil C.; PEVZNER, Pavel A.: „An Introduction to Bioinformatics Algorithms“. Erste Auflage. Bradford Book/MIT Press, 2004

[20] KORF, Ian; YANDELL, Mark; BEDELL Joseph: „BLAST. Basic Local Alignment Search Tool“. Erste Auflage. O’Reilly Media, 2003

[21] RÖDL, Marina: „BLAST (Basic Local Alignment Search Tool)“. Fakultät für Informatik, Universität Ulm, SS06
http://theorie.informatik.uni-ulm.de/Lehre/SS6/seminar_bioinf/blast_folien

[22] HANKELN, Thomas; LIEB, Bernd: „Genomforschung und Sequenzanalyse - Einführung in Methoden der Bioinformatik - “.
Johannes Gutenberg Universität Mainz, WS06/07
http://molgen.biologie.uni-mainz.de/Downloads/PDFs/Genomforsch/HVseqanal5-2006.pdf

[23] CHENNA, Ramu; SUGAWARA, Hideaki; KOIKE, Tadashi; LOPEZ, Rodrigo; GIBSON, Toby J.; HIGGINS, Desmond G.; Thompson, Julie D.: „Multiple sequence alignment with the Clustal series of programs“. Journal: Nucleic Acids Research 31, 2003
http://nar.oxfordjournals.org/cgi/content/abstract/31/13/3497

[24] Webpräsenz von EBI: CLUSTALW
http://www.ebi.ac.uk/clustalw/

[25] Online Enzyklopädie; Suchbegriff: „Clustal“
http://en.wikipedia.org/wiki/ClustalW

[26] BRAUER, German: „DIALIGN“.
Institut für Informatik, Humboldt-Universität zu Berlin
http://www2.informatik.hu-berlin.de/.../folien/MSA_DIAlign.ppt#0

[27] Dr. KÖNIG, Rainer: „Sequenz – Alignment Teil 2“.
Deutsches Krebsforschungszentrum Heidelberg, 2003
http://www.dkfz.de/ibios_old/lectures/bi1_ws0304/Multi141103_Mobitec.pdf

[28] MORGENSTERN, Burkhard: „DIALIGN 2: improvement of the segment-to-segment approach to multiple sequence alignment“.
Journal: Bioinformatics 15, Nr. 3, 1999
http://bioinformatics.oxfordjournals.org/cgi/reprint/15/3/211

[29] Dr. EILS, Roland: „Multiples-Sequenz-Alignment“.
Deutsches Krebsforschungszentrum Heidelberg, WS04/05
http://www.dkfz.de/ibios_old/lectures/bioinf_ws0405/reils/bioinfl0405.seq4.pdf

[30] GRASSO, Catherine; LEE, Christopher: „Combining partial order alignment and progressive multiple sequence alignment increases alignment speed and scalability to very large alignment problems“.
Journal: Bioinformatics 20, Nr. 10, 2004
http://bioinformatics.oxfordjournals.org/cgi/screenpdf/20/10/1546

[31] Prof. Dr. HAMMER, Barbara: „Algorithmische Bioinformatik“.
Institut für Informatik, Technische Universität Clausthal
http://www2.in.tu-clausthal.de/~hammer/lectures/bioinf/phylogenie.pdf

[32] MEIER, Andreas: „Relationale Datenbanken, Leitfaden für die Praxis“. Fünfte Auflage. Springer-Verlag Berlin · Heidelberg · New York, 2004

[33] Online Enzyklopädie; Suchbegriff: „Relationale Datenbank“
http://de.wikipedia.org/wiki/Relationale_Datenbank

[34] Prof. Dr. SEEWALDT, Thomas: Skript zur Vorlesung „Datenbanken“. Fakultät für Informationstechnologie, Hochschule Mannheim, SS06

[35] MATTHIESSEN, Günter; UNTERSTEIN, Michael: „Relationale Datenbanken und SQL: Konzepte der Entwicklung und Anwendung“. Erste Auflage. Addison-Wesley, 1997

[36] Online Enzyklopädie; Suchbegriff: „SQL“
http://de.wikipedia.org/wiki/Sql

[37] Webpräsenz von MySQL
http://dev.mysql.com/doc/refman/5.1/de/table-size.html

[38] Webpräsenz von Sun: JDBC
http://java.sun.com/docs/books/tutorial/jdbc/basics/gettingstarted.html

[39] Webpräsenz der University of Dundee: JalView
http://www.jalview.org/

[40] Webpräsenz des Department of Genome Science, University of Washington, Seattle
http://evolution.genetics.washington.edu/phylip/newicktree.html

[41] Webpräsenz der Universität Bielefeld: GenDB
http://www.cebitec.uni-bielefeld.de/groups/brf/software/gendb_info/

[42] Webpräsenz des MPI: Projekte
http://www.mpi-bremen.de/en/Projects.html

[43] HANEKAMP, Kristian: „Entwicklung eines Systems zur phylogenetischen Analyse von Sequenzvergleichen im Rahmen von Genom-annotationsprojekten“. Diplomarbeit

Anhang C: Abkürzungsverzeichnis

API ..... Application Programmer Interface
ATV ..... A Tree Viewer
AWI ..... Alfred-Wegener-Institut
BLAST ..... Basic Local ALignment Search Tool
BLOSUM ..... Blocks Substitution Matrix
DB ..... Datenbank
Dialign ..... Diagonal Alignment
DNA ..... Desoxyribonukleinsäure (engl. Deoxyribonucleic Acid)
EBI ..... European Bioinformatics Institute
EMBL ..... European Molecular Biology Laboratory
FASTA ..... Fast Alignment
GNU ..... GNU is not Unix
GPL ..... GNU General Public License
GUI ..... Graphical User Interface
HGF ..... Helmholtz Gemeinschaft Deutscher Forschungszentren
http ..... Hypertext Transfer Protocol
IT ..... Informationstechnologie
JalView ..... Java Multiple Alignment Viewer
JDBC ..... Java Database Connectivity
MPI ..... Max-Planck-Institut
mRNA ..... messenger RNA
NCBI ..... National Center for Biotechnology Information (USA)
ORF ..... offener Leserahmen (engl. Open Reading Frame)
PAM ..... Percent Accepted Mutation
PIR ..... Protein Information Resource (Georgetown University)
POA ..... Partial Order Alignment
RNA ..... Ribonukleinsäure (engl. Ribonucleic Acid)
SIB ..... Swiss Institute of Bioinformatics
SQL ..... Structured Query Language
SRS ..... Sequence Retrieval System

TrEMBL ..... Translated EMBL
UML ..... Unified Modeling Language
UniProt ..... Universal Protein Resource
UPGMA ..... Unweighted Pair Group with Arithmetic Mean
UTR ..... Untranslated Region
XML ..... Extensible Markup Language

Anhang D: Quellcode Datenbankerstellung

```
import java.sql.*;

public class cg_zero_init {

    static Connection      conn   = null;
    static Statement       st     = null;
    static public String   url    = "jdbc:mysql://.../cg_zero";
    static public String   user   = "user";
    static public String   pw     = "password";

    public static void main(String[] args) throws Exception {

        //  ------------------------------------------------------------------------------------------
        //  Verbindungsaufbau zur Datenbank
        //  ------------------------------------------------------------------------------------------
        try {
            Class.forName("com.mysql.jdbc.Driver").newInstance();
            conn = DriverManager.getConnection(url, user, pw);
            System.out.println("connected");
        }
        catch (Exception e) {
            System.err.println("no connection possible");
        }

        try {

        //  ------------------------------------------------------------------------------------------
        //  Tabelle „blast“ speichert die statistischen Daten der Gensequenzen, die
        //  während eines Blastprozesses lokalisiert werden.
        //  ------------------------------------------------------------------------------------------
        String blastSQL = "CREATE TABLE IF NOT EXISTS blast                    " +
        "    (queryName        VARCHAR(300) NOT NULL,                          " +
        "     geneID           VARCHAR(50),                                    " +
        "     eValue           FLOAT(20,10) NOT NULL,                          " +
        "     species          VARCHAR(800) NOT NULL,                          " +
        "     description      VARCHAR(800) NOT NULL,                          " +
        "     geneName         VARCHAR(20)  NOT NULL,                          " +
        "     queryStrand      INTEGER(20),                                    " +
        "     queryStart       INTEGER(20)  NOT NULL DEFAULT '0',              " +
        "     queryEnd         INTEGER(20)  NOT NULL DEFAULT '0',              " +
        "     subjectStart     INTEGER(20)  NOT NULL DEFAULT '0',              " +
        "     subjectEnd       INTEGER(20)  NOT NULL DEFAULT '0',              " +
        "     blastID          INTEGER(20),                                    " +
        "    PRIMARY KEY (geneID, blastID));                                   ";
```

```
// ---------------------------------------------------------------------------------------------
// Tabelle „blast_result“ repräsentiert die Gensequenzen, die für das
// Alignment selektiert wurden.
// ---------------------------------------------------------------------------------------------
String blast_resultSQL = "CREATE TABLE IF NOT EXISTS blast_result      " +
"    (geneID            VARCHAR(50),                                  " +
"     selected          INTEGER(2) NOT NULL,                          " +
"     blastID           INTEGER(20),                                  " +
"    FOREIGN KEY (geneID) REFERENCES blast(geneID),                  " +
"    FOREIGN KEY (blastID) REFERENCES blast(blastID));               ";

// ---------------------------------------------------------------------------------------------
// Tabelle „alignment“ speichert die, mit dem Query übereinstimmenden,
// Sequenzabschnitte der jeweiligen Gensequnezen.
// ---------------------------------------------------------------------------------------------
String alignmentSQL = "CREATE TABLE IF NOT EXISTS alignment      " +
"    (geneID            VARCHAR(4000),                                " +
"     seqCode           VARCHAR(4000) NOT NULL,                       " +
"     blastID           INTEGER(20),                                  " +
"    FOREIGN KEY (geneID) REFERENCES blast(geneID),                  " +
"    FOREIGN KEY (blastID) REFERENCES blast(blastID));               ";

// ---------------------------------------------------------------------------------------------
// Tabelle „tree“ speichert den Code zur Generierung des phylogenetischen
// Baums.
// ---------------------------------------------------------------------------------------------
String treeSQL = "CREATE TABLE IF NOT EXISTS tree                  " +
"    (nhCode            VARCHAR(2500),                                " +
"     blastID           INTEGER(20),                                  " +
"    FOREIGN KEY (blastID) REFERENCES blast(blastID));               ";

// ---------------------------------------------------------------------------------------------
// Tabelle „config“ speichert die Konfigurationsparameter, mit deren Hilfe
// die Blastabfrage initialisiert wird.
// ---------------------------------------------------------------------------------------------
String configSQL = "CREATE TABLE IF NOT EXISTS config              " +
"    (evalueCutoff      VARCHAR(50) NOT NULL,                         " +
"     alignMethod       VARCHAR(50) NOT NULL,                         " +
"     blastRule         VARCHAR(50) NOT NULL,                         " +
"     selectionRule     VARCHAR(50) NOT NULL,                         " +
"     treeMethod        VARCHAR(50) NOT NULL,                         " +
"     maxSequences      VARCHAR(50) NOT NULL,                         " +
"     analysisName      VARCHAR(50) NOT NULL,                         " +
"     blastID           INTEGER(20),                                  " +
"    FOREIGN KEY (blastID) REFERENCES blast(blastID));               ";
```

```
        // ---------------------------------------------------------------------------------------
        // Tabelle „neighbor“ speichert die evolutionäre Distanz jeder Gensequenz
        // zu dem gesuchten Query
        // ---------------------------------------------------------------------------------------
        String neighborSQL = "CREATE TABLE IF NOT EXISTS neighbor          " +
        "    (geneID          VARCHAR(50),                                 " +
        "     distance        FLOAT(20,10) NOT NULL,                       " +
        "     blastID         INTEGER(20),                                 " +
        "    FOREIGN KEY (geneID) REFERENCES blast(geneID),                " +
        "    FOREIGN KEY (blastID) REFERENCES blast(blastID));             ";

        // ---------------------------------------------------------------------------------------
        // Tabelle „minNeighbor“ speichert die Distanz und die Gensequenz, die
        // den evolutionär geringsten Abstand zum gesuchten Query besitzt.
        // ---------------------------------------------------------------------------------------
        String minNeighborSQL = "CREATE TABLE IF NOT EXISTS minNeighbor    " +
        "    (minGeneID       VARCHAR(50),                                 " +
        "     minDistance     FLOAT(20,10) NOT NULL,                       " +
        "     blastID         INTEGER(20),                                 " +
        "    FOREIGN KEY (blastID) REFERENCES blast(blastID));             ";

        // ---------------------------------------------------------------------------------------
        // Tabelle „sequence“ repräsentiert eine Beziehungstabelle, um der
        // betreffenden Sequenznummer die richtige Gensequenz zuzuweisen.
        // ---------------------------------------------------------------------------------------
        String sequenceSQL  = "CREATE TABLE IF NOT EXISTS sequence         " +
        "    (seqNo           VARCHAR(50) NOT NULL,                        " +
        "     seqName         VARCHAR(50) NOT NULL,                        " +
        "     blastID         INTEGER(20),                                 " +
        “     FOREIGN KEY (blastID) REFERENCES blast(blastID));             ";

        st = conn.createStatement();
        st.execute(blastSQL);
        st.execute(blast_resultSQL);
        st.execute(alignmentSQL);
        st.execute(treeSQL);
        st.execute(configSQL);
        st.execute(neighborSQL);
        st.execute(minNeighborSQL);
        st.execute(sequenceSQL);
            }
        }
            catch (Exception e) {
                e.printStackTrace();
                System.err.println("error in database"); }
        finally { if (conn != null) { try { conn.close(); }
                catch (Exception e) {  }
            }
        }
    }
}
```