Hochschule Karlsruhe
Technik und Wirtschaft
UNIVERSITY OF APPLIED SCIENCES

Diplomarbeit

im Studiengang Kartographie und Geomatik

Entwicklung eines Konvertierungswerkzeugs zur Übernahme von Daten der § 24a-Biotopkartierung in das Arteninformationssystem (ARTIS) und Aufbau eines Internet-Kartendienstes zur Visualisierung der Artenfunde

Nadine Kastner

15.12.2006

Erstellt an der Landesanstalt für Umwelt,
Messungen und Naturschutz Baden-Württemberg

HOCHSCHULE KARLSRUHE –TECHNIK UND WIRTSCHAFT
Fakultät Geomatik – Studiengang Kartographie und Geomatik

Diplomarbeit

für Frau Nadine K a s t n e r

Thema: **Entwicklung eines Konvertierungswerkzeugs zur Übernahme von Daten der § 24a-Biotopkartierung in das Arteninformationssystem (ARTIS) und Aufbau eines Internet-Kartendienstes zur Visualisierung der Artenfunde**

Die Landesanstalt für Umwelt, Messungen und Naturschutz Baden-Württemberg (LUBW) hat in den letzten Jahren mehrere Anläufe unternommen um im Bereich Artenschutz, der einen Bioindikator und wichtigen Bestandteil des Naturschutzes darstellt, einen einheitlichen Gesamtdatenbestand zu Arten und Artenfunden zu erzielen. Wichtige Schritte in Richtung einer zentralisierten Artendatenbank wurden bereits durch die Erstellung der Software „Artenerfassung“ und die Standardisierung des Artenlexikons erzielt.
Ziel eines Arteninformationssystems ist die Zentralisierung und Optimierung der Nutzung der Artdaten, so dass letztlich ein gemeinsamer Artenbestand mit Abfrage- und Ausgabemöglichkeit entsteht. Kerndaten des Systems sind die Angaben bezüglich Arten (Artenfund) und ihrer geographischen Komponente (Fundort). Ebenfalls von großer Bedeutung sind die Informationen über den Zeitpunkt der Beobachtung und den Erfasser (Kartierer).

Zum jetzigen Zeitpunkt beinhaltet ARTIS nur die Daten der FLOREIN-Kartierung des Bundesamtes für Naturschutz (BfN, Bonn). Im Rahmen der Diplomarbeit soll zunächst ein Konzept zur Überführung der vorliegenden Artdatenbestände aus Flächenkartierungen und Einzelfunden in Baden-Württemberg in ARTIS entwickelt werden. Dieses Konzept ist im Anschluss beispielhaft anhand der Daten der Biotopkartierung nach § 24a Naturschutzgesetz (NatSchG) programmtechnisch zu realisieren.
Mit ca. 1,8 Millionen Artenfunden von etwa 6000 verschiedenen Arten ist der Datenbestand der § 24a-Biotopkartierung der größte systematische Artdatenbestand Baden-Württembergs. Die Ergebnisse dieser Kartierung bilden eine der wichtigsten Grundlagen für den Naturschutz, die Landschaftsplanung und zur Beobachtung der Landschaftsentwicklung.

Ziel des ersten Teils der Diplomarbeit ist es, das vorhandene Datenmodell weiterzuentwickeln und durch die Formulierung einer Schnittstelle und Programmierung eines Konverters die Daten der Biotopkartierung in das Schema ARTIS zu übertragen.

Der zweite Teil der Arbeit befasst sich mit den Möglichkeiten zur Visualisierung der Artdaten Baden-Württembergs. Die Aufgabe trägt zur Erfüllung der Vorschriften des Umweltinformationsgesetz (UIG), Fassung vom 22.12.2004, bei. Dieses verpflichtet die informationsführenden Stellen, der Öffentlichkeit Einsicht in umweltrelevante Daten zu ermöglichen, beispielsweise über das Internet. Aus dieser Vorgabe heraus soll im Rahmen der Diplomarbeit ein Kartendienst erstellt werden, der in dynamischer Form das Vorkommen der einzelnen Arten darstellt.

Für die Diplomarbeit stehen die landesweiten Datenbestände des Umweltinformationssystems (UIS) zur Verfügung, die in einer Oracle 10g Datenbank abgelegt sind. Die an der LUBW vorhandenen Organisationswerkzeuge und Software-Frameworks (ArcGIS 9, etc.) können zur Ausführung der Arbeit genutzt werden.

Bearbeitungszeit: 4 Monate

Ausgabedatum: 15. August 2006
Abgabetermin: 15. Dezember 2006

…….
(Prof. Dr. rer. nat. Detlef Günther-Diringer)

…….
(Dipl. Biol. Ulrich Hönig)

Erklärung

Hiermit versichere ich, dass ich diese Diplomarbeit selbständig verfasst und außer den von mir angegebenen keine anderen Quellen und Hilfsmittel verwendet habe.

Karlsruhe, den 15.12.2006

.....
Nadine Kastner

Danksagung

An dieser Stelle möchte ich mich bei allen bedanken, die zum Gelingen dieser Arbeit beigetragen haben.

Mein besonderer Dank gilt:

- Prof. Dr. rer. nat. Detlef Günther-Diringer, für die Betreuung der Arbeit von Seiten der Hochschule
- Manfred Müller und Wolfgang Schillinger, die mir die Anfertigung der Arbeit im ITZ ermöglicht haben
- meinem Betreuer Ulrich Hönig, der mir mit Rat und Tat zur Seite gestanden hat und diese Arbeit korrigiert hat
- Vicente Aguayo, der immer ein offenes Ohr für mich hatte und mir mit so manchem Schokoriegel die Arbeit versüßt hat
- Frau Weiss und Frau Fett, für die nette Aufnahme in ihrem Büro
- Frank Eberspächer, für das Korrekturlesen und die vielen hilfreichen Tips
- allen Mitarbeitern des Sachgebiets 53.2 (ITZ), besonders Martin Scherrer, Jörg Strittmatter und Bastian Ellmenreich für die wertvollen Anregungen und die fachliche Unterstützung

Besonders bedanken möchte ich mich außerdem bei:

- meinen Eltern, die mich immer unterstützt, mir in stressigen Zeiten den Rücken freigehalten und mir einfach so viel in meinem Leben ermöglicht haben
- meinen Freund Florian, der mit mir durch Dick und Dünn gegangen ist. Danke für deine unendliche Liebe und Geduld!
- meinen Großeltern, die mir durch kleine oder größere finanzielle Gaben das Leben erleichert haben und immer ein paar aufbauende Worte parat hatten

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Der Artenschutz als wichtiger Teil des Naturschutzes ist ein ebenso komplexer wie interessanter Bereich, der in den letzten Jahren gerade in Baden-Württemberg mehr und mehr an Bedeutung gewonnen hat. Um die Artenvielfalt auch für zukünftige Generationen zu erhalten, muss neben der Sicherung der Lebensräume auch die Information der Bevölkerung weiter vorangetrieben werden. Denn nur wenn jeder einzelne Bürger sich seiner Verantwortung für eine intakte Umwelt bewusst ist, können Tier- und Pflanzenarten nachhaltig geschützt werden. Eine abwechslungsreich gestaltete Kulturlandschaft ist nicht zuletzt auch aufgrund ihrer Funktion als Erholungsraum für den Menschen und zur Steigerung der Lebensqualität jedes Einzelnen von unschätzbarem Wert.

In den letzten Jahrzehnten wurden auf dem Gebiet des Naturschutzes zahlreiche wertvolle Datenquellen angelegt. Die Zusammenführung dieser Naturschutzdaten ist eine wichtige und große Aufgabe, die in vielen Bereichen auch schon verwirklicht wurde. In Baden-Württemberg steht im Bereich Artenschutz aus verschiedenen Gründen die Zusammenführung der Datenbestände noch aus. Diese Diplomarbeit soll dazu beitragen, den Prozess voranzutreiben und Anregungen für Realisierungsmöglichkeiten der Datenkonsilidierung und -visualisierung geben.
Bauer und Günzel ([BG04]) formulieren die Probleme bei der Integration verschiedener Datenbestände in ein System wie folgt: *„Der Vorgang der Vereinigung von Daten aus autonomen Datenbeständen ist besonders schwierig, wenn heterogene Daten unterschiedlicher Qualität, in verschiedenen Datenformaten, in heterogenen Datenmodellen und Datenbanksystemen gehalten werden.“*
Um ein besseres Verständnis für die Organisation des Natur- und damit auch des Artenschutzes zu vermitteln, soll zunächst der Aufbau der Naturschutzverwaltung in Baden-Württemberg erläutert werden.

1.1 Die Naturschutzverwaltung in Baden-Württemberg

Die Naturschutzverwaltung in Baden-Württemberg kann in drei Verwaltungsebenen unterteilt werden. Dies sind die Landes-, die Regierungsbezirks- und die Kreisebene. Auf der Landesebene befindet sich mit dem Ministerium für Ernährung und Ländlichen Raum (MLR) die Oberste Naturschutzbehörde. Diese wird in fachlichen Angelegenheiten von der Landesanstalt für Umwelt, Messungen und Naturschutz (LUBW) beraten. Im Gegenzug führt das Ministerium die Fachaufsicht über die LUBW sowie die vier Regierungspräsidien, welche die mittlere Verwaltungsebene bilden.
Die unteren Naturschutzbehörden befinden sich in den 35 Landrats- und 9 Bürgermeisterämtern der Stadtkreise, über die wiederum die Regierungspräsidien die Fachaufsicht führen. Beraten werden die unteren Naturschutzbehörden von ca. 220 Naturschutzbeauftragten und ehrenamtlichen Beratern. Zur unteren Verwaltungsebene zählen nach § 14 Landesverwaltungsgesetz außerdem die unteren Naturschutzbehörden bei den Großen Kreisstädten und verein-

barten Verwaltungsgemeinschaften, die jedoch nur eine eingeschränkte Zuständigkeit besitzen. Wie in Abbildung 1.1 erkennbar, ergänzen sich Verwaltungsebene (linke Spalte) und fachliche Beratungsebene (rechte Spalte). Die Behörden der fachlichen Beratungsebene können aufgrund ihrer beratenden Funktion keine Verordnungen erlassen oder Rechtsbescheide erteilen, aber ihr Veto gegen Entscheidungen der Verwaltungsbehörden einlegen. Erwähnt werden müssen auch die vielen ehrenamtlichen Mitarbeiter, die z.B. als Naturschutzwarte in ihrer Freizeit interessierte Bürger durch die Natur führen, über Bestimmungen des Naturschutzes informieren und so erheblich zum besseren Verständnis für den Naturschutz in der Bevölkerung beitragen. Nur wenn das Zusammenspiel der einzelnen Behörden mit den beratenden Stellen funktioniert, kann der Naturschutz effektiv vorangebracht werden.

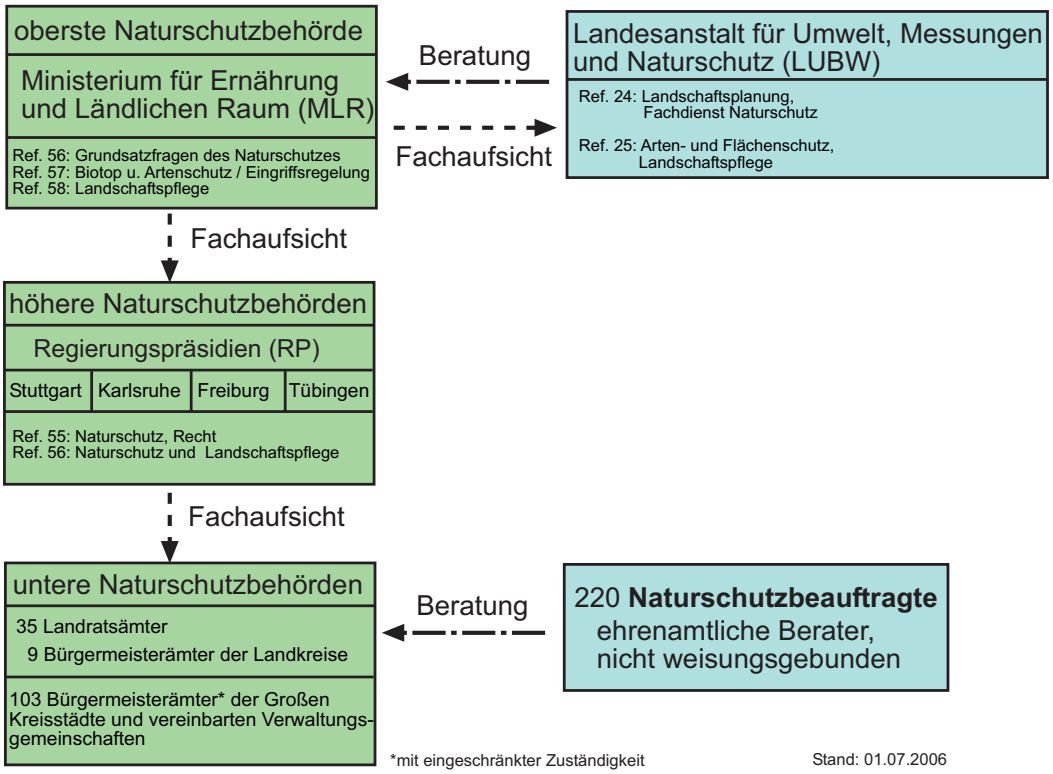


Abbildung 1.1: Die Staatliche Naturschutzverwaltung Baden-Württemberg (nach [MLR06])

Die Hauptaufgaben der LUBW liegen in der Beratung und Unterstützung der Landesregierung und der Landesbehörden, der Umweltbeobachtung, der Beurteilung umweltrelevanter Sachverhalte und der Information der Öffentlichkeit. Durch die Novellierung des Umweltinformationsgesetzes (UIG) im Dezember 2004 gewinnt die Aufgabe der Versorgung der Öffentlichkeit mit umweltrelevanten Daten und Informationen mehr denn je an Bedeutung. Das UIG (Fassung vom 22.12.2004) schafft dabei den rechtlichen Rahmen für den freien Zugang zu den Umweltinformationen der informationspflichtigen Stellen und für die Verbreitung dieser Informationen. Als informationspflichtige Stellen werden in § 2 Abs. 1 „*[...] die Regierung und andere Stellen der öffentlichen Verwaltung*“ genannt. „*Gremien, die diese Stellen beraten, gelten als Teil der Stelle, die deren Mitglieder beruft[...]*“ [UIG04].

1.2 Informationsangebote zum Themenbereich Naturschutz im World Wide Web

Neben den unterschiedlichen Informationsangeboten, die auf der Webseite der LUBW unter `http://www.lubw.baden-wuerttemberg.de` im Bereich „Infodienste“ zu finden sind, entwickelt das Informationstechnische Zentrum (ITZ) der LUBW in Karlsruhe in Zusammenarbeit mit dem MLR auch weitere Dienste und stellt diese der Allgemeinheit über verschiedene Seiten im Internet zur Verfügung. Den besten Überblick über die im Netz verfügbaren Informationen zu Natur-, Arten- und Umweltschutz bietet das Umweltportal Baden-Württemberg, zu dem man u.a. über einen Link in der Rubrik „Infodienste“ gelangt. Die dort vorhandenen Angebote wurden für die Diplomarbeit teilweise zur Beschaffung von Informationen genutzt, sollen an dieser Stelle jedoch nicht weiter erläutert werden. Einzig über das Angebot „Naturschutz-Fachinformationen im World Wide Web (NafaWeb)“ wird hier ein kurzer Überblick geliefert, da geplant ist, das dort angesiedelte Bildarchiv mit dem in der Diplomarbeit geschaffenen Kartendienst zu verknüpfen.

NafaWeb – Naturschutz-Fachinformationen im World Wide Web

Als Online Informationssystem betreibt der Naturschutz-Fachdienst der LUBW das NafaWeb. Dieses ist über die Webseite der LUBW über den Link XfaWeb, im Bereich „Internes Fachangebot der LUBW“ zugänglich. XfaWeb ist der Oberbegriff für verschiedene Web-basierte Informationssysteme. Zur XfaWeb-Familie zählen neben NafaWeb auch die Fachinformationssysteme AbfaWeb (Abfallbehandlung), AlfaWeb (Altlastenbearbeitung), BofaWeb (Bodenschutz), ChemfaWeb (Behördliches Chemikalienmanagement) und FofaWeb (Umweltforschung in Baden-Württemberg). Abbildung 1.2 zeigt die Startseite des NafaWeb.

Ziel dieses Portals ist es, Arbeitshilfen für Fachleute und Beauftragte der verschiedenen Verwaltungsebenen im Naturschutzbereich bereitzustellen und so die Koordination und Einheitlichkeit zu verbessern. Die vom Fachdienst Naturschutz bereitgestellten Informationen können dabei über modernste Informations- und Kommunikationsmittel abgerufen werden und ermöglichen so eine schnelle Informationsbeschaffung. Neben einem gegliederten Zugang bietet die Seite auch eine Schlagwort- und Volltextsuche und ein Glossar. Unter dem Punkt „Berichte“ können außerdem Berichte aus den zahlreichen Veröffentlichungen der Naturschutzverwaltung und viele weitere Informationen abgerufen werden.
Der für die Diplomarbeit interessanteste Bereich des Angebots ist unter dem Gliederungspunkt „Datenbanken“ zu finden. Er beinhaltet neben dem Bildarchiv auch das Abfragesystem für die § 32- (ehemals § 24a-)Biotope und das Reportsystem zur Erzeugung von Berichten über die Biotoptypen. Wählt man beispielsweise die Abfrage der § 32-Biotope aus, so erhält man nach Selektion einer Verwaltungseinheit aus der Liste eine Ergebnistabelle im EXCEL- oder HTML-Format, Karten zu den geschützten Biotopen in Punkt- und Flächendarstellung und Berichte in Form einer Biotopliste oder eines Erhebungsbogens. Nicht möglich ist es bisher, sich die in den Biotopen lebenden Arten ausgeben zu lassen. Im Rahmen der Diplomarbeit soll die Visualisierung von Artenfunden in einem Kartendienst, unabhängig von einem Biotop oder sonstigen Schutzgebiet, realisiert werden.
Im Bereich „Bildarchiv“ befinden sich ca. 22.500 digitale Bilder (Stand 08.08.2005), die im Laufe der Jahre von den Referaten 24 (Artenschutz, Fachdienst Naturschutz) und 25 (Flächenschutz, Landschaftsplanung und -pflege) der LUBW gesammelt wurden. Damit die Suche nach

Abbildung 1.2: Einstiegsseite des Online Informationssystems NafaWeb

Bildmaterial möglichst einfach ist, wurde das Archiv in fünf Themenkomplexe unterteilt. Es handelt sich hier um die Bereiche: Naturschutzgebiete, Landschaftsschutzgebiete, Biotoptypen, Lebensraumtypen sowie Tier- und Pflanzenarten. Die Einbindung des Bildarchivs in die im Rahmen der Diplomarbeit entstehende Kartendienstanwendung stellt eine sinnvolle zur Erweiterung des Informationsangebotes dar.

2 Ausgangssituation

2.1 Datenhaltung in der LUBW

Bei der Landesanstalt für Umwelt, Messungen und Naturschutz Baden-Württemberg werden Sach- und Geodaten in unterschiedlichen Systemen gehalten. Um eine grobe Übersicht über die Datenhaltung zu schaffen, soll hier zunächst das Umweltinformationssystem Baden-Württemberg als übergreifendes System näher erläutert werden.

2.1.1 Das Umweltinformationssystem Baden-Württemberg (UIS)

Um die komplexen Umweltprobleme in Baden-Württemberg lösen zu können, bedarf es der Fachkenntnis vieler Experten, die jedoch auf die unterschiedlichen Umweltbehörden verteilt sind. Gerade heutzutage müssen in verstärktem Maße die Zusammenhänge beachtet werden, die zwischen den unterschiedlichen Umweltbereichen bestehen. Aus diesem Grund wurde 1996 das Umweltinformationssystem Baden-Württemberg ins Leben gerufen, das die Daten und Erkenntnisse der Fachleute der unterschiedlichen Umweltbereiche zusammenführt und so den gemeinsamen Zugriff auf die unterschiedlichen Datenbestände erlaubt. In den letzten zehn Jahren wurde das UIS stetig um verschiedene Komponenten und Daten ergänzt.
Der Nutzer kann über unterschiedliche Fachanwendungen, die aus den Bereichen „Wasser, Immissionsschutz, Boden, Abfall und Arbeitsschutz“ (WIBAS) und „Naturschutz“ (NAIS) stammen, auf die Daten des UIS zugreifen. Aufgabe des UIS ist einerseits, die Umweltbehörden in ihrer Arbeit zu unterstützen, aber auch möglichst aktuelle Daten aus den verschiedenen Bereichen für Bürgerinnen und Bürger bereitzustellen. Unter Beachtung von Datenschutz und Datensicherheit, erhalten auch Privatpersonen über bestimmte im Internet angebotene Dienste Zugang zu den im UIS gehaltenen Daten. Damit erfüllt die LUBW die den Landesbehörden durch das novellierte Umweltinformationsgesetz (UIG vom 07. März 2006) auferlegte Aufgabe, der Öffentlichkeit über geeignete elektronische Medien Einblick in die Umweltdaten zu ermöglichen. Eine der frei zugänglichen Komponenten ist der dynamische Internet-Dienst *Umwelt-Datenbanken und -Karten online*, der ebenfalls in der Rubrik *Infodienste* unter `http://www.lubw.baden-wuerttemberg.de` zu finden ist.

Das Räumliche Informations- und Planungssystem (RIPS) dient der raumbezogenen Datenverarbeitung. Hauptaufgabe des RIPS ist es, die verschiedenen Aktivitäten zur Aufbereitung und Bereitstellung von Geodaten zu koordinieren und die Daten für den GIS-Einsatz nutzbar zu machen. Bereitgestellt und gewartet wird dieses Informationssystem vom Sachgebiet „Raumbezogene Informationssysteme“ im Referat 53 der LUBW. Der Gesamtdatenbestand, der auch als „RIPS-Pool“ bezeichnet wird, dient der Haltung von raumbezogenen Fach- und Basisdaten. Die im RIPS-Pool geführten Maßstabsbereiche können Abbildung 2.1 entnommen werden.
Um einen genaueren Überblick über die dort gespeicherten Daten zu bekommen, sollen diese nachfolgend kurz tabellarisch erläutert werden.

Maßstab	Bereich	Grundlage
M1	1:1 bis 1:10.000	Automatisierte Liegenschaftskarte (ALK)
M2	1:10.000 bis 1:50.000	Digitales Basis-Landschaftsmodell (Basis-DLM) des Amtlichen Topographisch-Kartographischen Informationssystems (ATKIS)
M3	1:50.000 bis 1:350.000	Topographische Übersichtskarte 1:200.000 (TÜK 200) bzw. das Digitale Landschaftsmodell 1:200.000 (DLM 200)
M4	1:350.000 bis 1:1.500.000	Übersichtskarten 1:500.000 oder 1:1.000.000 und gegebenenfalls auch kleinere Maßstäbe. Das räumliche Bezugssystem ist eine Kegelprojektion (Lambert)

Tabelle 2.1: Im RIPS-Pool geführte Maßstabsbereiche

Zunächst können die Daten grob in Geo- und Sachdaten unterteilt werden. Unter Geodaten versteht man allgemein digitale Daten im Raster- oder Vektorformat, die in unterschiedlichen Datenmodellen und Speicherformaten vorliegen. Grundsätzlich zählen zu den Geodaten auch alle punktförmigen Objekte wie Messstellen und Artenfundorte. Allerdings unterscheiden sich die Punktdaten in der fachlich-organisatorischen Bearbeitung wesentlich von den erstgenannten. Diese Daten besitzen als geometrisches Objektattribut die Koordinate, welche jedoch in den allermeisten Fällen direkt bei den Sachdaten gehalten wird. Deshalb unterliegen sie bezüglich Genauigkeit, Generalisierungsgrad etc. anderen Rahmenbedingungen als die übrigen Vektordaten.

Geo- und Sachdaten lassen sich jeweils wiederum in Fach- und Basisdaten unterscheiden. Als Basisdaten gelten die von der Vermessungsverwaltung geführten raumbezogenen Daten, die auszugsweise in Tabelle 2.2 aufgeführt sind.

Basisdaten	Maßstab
ALK (Amtliches Liegenschaftskataster)	1:500 bis 1:2.500
Digitale Orthophotos (DOP)	1:10.000
Digitale Übersichtskarte (ÜK 10)	1:1.000.000
Digitale Übersichtskarte (ÜK 500)	1: 500.000
Digitale Topographische Übersichtskarten (TÜK200)	1:500.000
Digitale Topographische Karten (TK25, TK50, TK100)	1:25.000, 1:50.000, 1:100.000
Digitales Höhenmodell (DHM 50)	Auflösung 50 m
Digitales Landschaftsmodell (ATKIS DLM 1000)	1:1.000.000
Digitales Landschaftsmodell (ATKIS DLM 25 BW)	1:10.000

Tabelle 2.2: Auszug aus den Basisdaten der Vermessungsverwaltung im RIPS-Pool

Als geometrische Fachdaten werden Fachdaten in Form von Raster- oder Vektordaten bezeichnet. In der klassischen Einteilung werden zwei Themenbereiche unterschieden. Es handelt sich um die Bereiche Naturschutz und Landschaftsökologie (Tabelle 2.3) sowie Wasser und Boden (Tabelle 2.4).

Naturschutz und Landschaftsökologie	Maßstab
Biotopkartierung nach §24a	1:25.000
NATURA 2000-Gebiete	1:25.000
Waldschutzgebiete (Bann- und Schonwälder)	1:25.000
Natur- und Landschaftsschutzgebiete	1:25.000, 1:250.000
Landschaftszerschneidung	1:25.000
Naturräumliche Gliederung	1:200.000

Tabelle 2.3: Auszug aus den Daten zu Naturschutz und Landschaftsökologie im RIPS-Pool

Wasser und Boden	Maßstab
Gewässernetz	1:10.000, 1:50.000, 1:200.000
Flussgebiete (Gewässereinzugsgebiete)	1:50.000
Wasser- und Quellschutzgebiete	1:25.000
Grundwasserlandschaften	1:200.000
Bodenübersichtskarte (BÜK200)	1:200.000
Geotope, Bodendenkmale	1:25.000
Bodenkarte	1:25.000, 1:50.000
Moore	1:5.000, 1:25.000

Tabelle 2.4: Auszug aus den Daten zu Wasser und Boden im RIPS-Pool

2.2 Das Arteninformationssystem ARTIS

In den letzten Jahren wurden im Bereich Artenschutz mehrere Anläufe unternommen, einen einheitlichen Geo- und Sachdatenbestand zu erzielen. Die Standardisierung des Artenlexikons und die Entwicklung des Artenerfassungsprogramms waren wichtige Schritte in Richtung der Erstellung einer zentralisierten Artdatenbank. Ziel eines Arteninformationssystems ist letztendlich, die Daten zentral in einem System mit Abfrage und Ausgabemöglichkeit zu halten, das die Nutzung der Artdaten optimiert. Basis dieses neuen Informationssystems ist das Datenbankschema ARTIS, das in einer Oracle Datenbank der LUBW in Karlsruhe geführt wird. Weiterführende Informationen zu diesem Schema sind in Kapitel 2.2.1 zu finden.
Kerndaten des Arteninformationssystems sind die Angaben bezüglich Arten (Artenfund) und ihrer geographischen Komponente (Fundort). Ebenfalls von großer Bedeutung sind die Informationen über den Zeitpunkt der Beobachtung, den Erfasser und die Herkunft der Daten. Werden also Daten aus verschiedenen Datenquellen in ein System überführt, gilt es darauf

zu achten, dass mindestens diese Angaben, im Regelfall aber auch weitere Angaben zur Art, importiert werden. Bisher liegen die sehr heterogenen Artdatenbestände in verschiedenen Formaten und Systemen gespeichert vor. Um diese Artdaten in ein System zusammenführen zu können, bedarf es einer aus fachlichen und technischen Gesichtspunkten gut durchdachten Datenmodellierung. Bereits Anfang 2004 wurde mit der Erarbeitung eines Konzepts für ein komplexes Arteninformationssystem an der LUBW begonnen.

2.2.1 Überblick über das Datenbankschema ARTIS

Mit der Modellierung des Datenbankschemas ARTIS wurde bereits vor etwa zwei Jahren begonnen. Die Struktur dieses neuen Schemas lehnt sich an die des Artenerfassungsprogramms an, da die meisten in Zukunft zu erwartenden Erhebungen auf den Standards des Artenerfassungsprogramms basieren werden. Nähere Informationen zum Artenerfassungsprogramm sind in Kapitel 2.2.2 zu finden.
Anfang 2005 wurden bereits die Daten der FLOREIN-Kartierung in das Schema importiert. Dieses besteht aus 21 Tabellen mit insgesamt 181 Spalten. Um sich einen ersten Überblick über das Schema zu verschaffen, wurde zunächst ein Entity-Relationship (ER)-Diagramm erzeugt. Dieses bildet grafisch die Tabellen mit den darin enthaltenen Feldern und den Datentypen der Felder ab und zeigt die Beziehungen zwischen den Tabellen durch Verbindungslinien an. In Anhang D der Diplomarbeit ist das ER-Diagramm von ARTIS zu finden.
Prinzipiell können alle Tabellen des Schema ARTIS für den Import der Artdaten verwendet werden. Welche Tabellen sich jedoch für die Speicherung der zu importierenden Daten eignen, musste erst herausgefunden werden. Eine Aussage darüber, welche Spalten des Schemas ARTIS nun tatsächlich mit Daten der § 24a-Biotopkartierung gefüllt werden sollen, war erst nach eingehender Analyse des Quellschemas ALBIS möglich. Die ausführliche Beschreibung dieses Schemas ist in Kapitel 2.3 zu finden.

2.2.2 Quellen für Artdaten/Artdatenbestände

Die Artdaten in Baden-Württemberg liegen in verschiedenen Systemen mit unterschiedlicher Struktur vor. In diesem Kapitel soll kurz auf die Datenquellen eingegangen werden, um deutlich zu machen, wo die Schwierigkeiten bei der Schaffung eines einheitlichen Gesamtdatenbestands liegen. Die nachfolgend erläuterten Datenbestände sind diejenigen, die bis jetzt für den Import in ARTIS in Erwägung gezogen wurden. Es ist jedoch wahrscheinlich, dass in den folgenden Jahren weitere umfangreiche Datenquellen hinzukommen werden.

Natura 2000

Unter Natura 2000 versteht man die von den Staaten der europäischen Union beschlossene Naturschutzkonzeption zur Erhaltung der biologischen Vielfalt in Europa. Um das europäische Naturerbe auch für künftige Generationen zu erhalten, wurde bereits 1992 die FFH-Richtlinie (Fauna-Flora-Habitat; Richtlinie 92/43/EWG des europäischen Rates vom 21.05.1992) herausgegeben, die besagt, dass unterschiedliche natürliche und naturnahe Lebensräume und Vorkommen gefährdeter Tier- und Pflanzenarten über verschiedene geografische Regionen hinweg miteinander zu verknüpfen sind. Zusammen mit den Gebieten der bereits 1979 erlassenen EU-Vogelschutzrichtlinie (SPA = Special Protected Area) bilden sie das Schutzgebietsverbundsystem NATURA 2000.

Die wichtigsten Artdatenquellen der Natura 2000-Gebiete sind die in Art. 6 Abs. 1 der FFH-Richtlinie definierten Pflege- und Entwicklungspläne (PEPL), die als Grundlage zum Schutz und Erhalt der Gebiete dienen. Für jedes Natura 2000-Gebiet soll daher in den nächsten Jahren ein Pflege- und Entwicklungsplan erstellt werden. In Zusammenarbeit von Ref. 25 und Ref. 53-ITZ entstand auf Basis von MS Access und ArcView GIS (ESRI) die PEPL-Software. Diese bildet die Grundlage für die Aufnahme von parzellenscharf abgegrenzten und bewerteten Lebensraumtypen, von Vorkommen der Lebensstätten der verschiedenen Arten sowie von erforderlichen Erhaltungs- und Entwicklungsmaßnahmen.
FFH- und SPA-Gebiete sind bereits in Datenbanken abgelegt. Die zu diesen Gebieten über die PEPL-Software eingegeben Daten werden derzeit noch in lokalen Access Datenbanken gespeichert. Zukünftig sollen diese Daten, inklusive der Artdaten, in die zentrale Oracle Datenbank übernommen werden. [Lan06b]

Das Fachinformationssystem Naturschutz (FIS-Natur)

Das Anwendungsmodul „Fachinformationssystem Naturschutz“ (FIS-Natur) ist eine in der Entwicklungsumgebung Centura 3.1.5 entwickelte Datenbankanwendung, die über eine Datenhaltung in Oracle verfügt. Bei diesem Programm handelt es sich um eine Fachanwendung, die zur Bearbeitung von Schutzgebieten wie Naturschutzgebieten (NSG), Landschaftsschutzgebieten (LSG), Flächenhaften Naturdenkmalen (FND) und Natura 2000-Gebieten eingesetzt wird. Alle im Programm verwendeten Datenschlüssel basieren auf den einheitlichen Naturschutz-Datenschlüsseln der LUBW.
FIS-Natur ist mit der in Abbildung 2.1 dargestellten GIS-Anwendung RIPS-Viewer verknüpft. Der RIPS-Viewer ist ein einfach zu bedienendes GIS-Werkzeug zur Visualisierung von Geodaten, wie z.B. Naturschutz- und Landschaftsschutzgebieten. Es wurde vom Informationstechnischen Zentrum der LUBW entwickelt und ist lizenzkostenfrei im Rahmen des Umweltinformationssystems Baden-Württemberg einsetzbar. Neben der Anzeige von Gebieten ist auch die Erfassung von Koordinaten (z.B. bei Artenfundorten) und Polygonen unter Verwendung von hinterlegten topographischen Karten, Orthophotos und ALK-Daten (Automatisierte Liegenschaftskarte), möglich.

Das Modul verfügt des Weiteren über eine Verknüpfung mit dem Dokumenten-Viewer, der die Anzeige und Zuordnung von Dokumenten zu den verschiedenen Schutzgebieten in allen gebräuchlichen Text-, Grafik-, Video- und Audioformaten erlaubt. Aufgrund einer unveränderbaren Kern-Datenstruktur kann das Programm auf alle Dokumente der verschiedenen WIBAS-Module einheitlich zugreifen. Aktuell wird FIS-Natur in der Version 3.6.0 eingesetzt.

Da es in der Diplomarbeit schwerpunktmäßig um die Speicherung und Darstellung von Artenfunden geht, soll hier kurz der Teilbereich „Artenerfassung“ der Anwendung FIS-Natur erläutert werden.
Die Erfassung von Artenfunden in Schutzgebieten und Pflegeflächen ist über das Erfassungsmodul in einfacher Form möglich. Es erlaubt die Kartierung und Aufnahme von ca. 15 Artengruppen (Höhere Pflanzen, Flechten, Vögel, Fledermäuse, etc.). Im Erfassungsformular sind folgende Datenfelder einzugeben: Kurzbezeichnung, Deutscher Name, Wissenschaftlicher Name, Arttyp (wird nach Eingabe des Namens automatisch eingetragen), Art der Häufigkeit, Häufigkeit, Status, Erfassungsdatum, Recht- und Hochwert, Bemerkung und Kartierer. Diese Daten können nur einmal pro Schutzgebiet unter der zugehörigen eindeutigen Schutzgebiets-

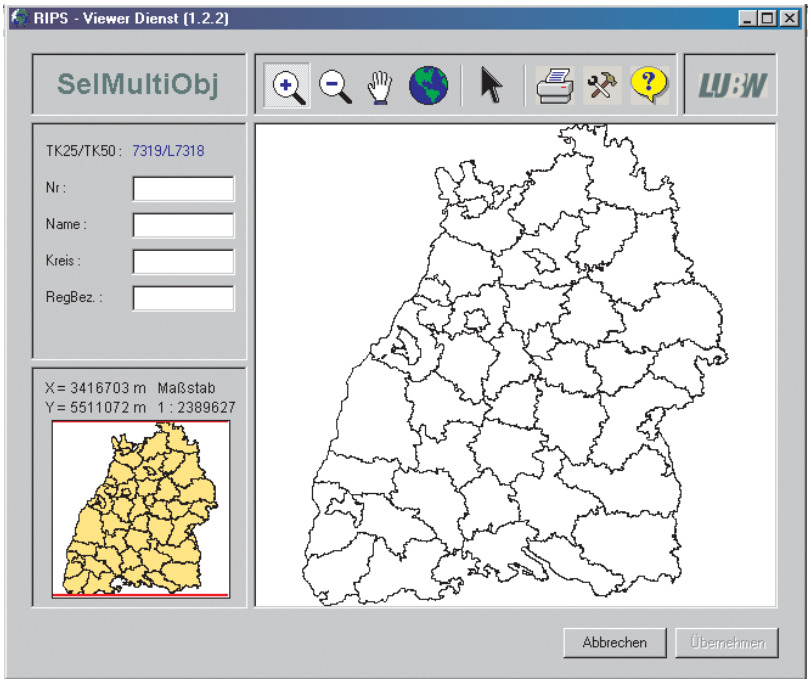


Abbildung 2.1: Das GIS-Werkzeug RIPS-Viewer

nummer abgelegt werden.
Neben den vom Benutzer eingegeben Daten werden noch weitere Informationen gespeichert. Durch die Eingabe des Schutzgebietsnamens wird in der Datenbank automatisch das Feld für die Schutzgebietsnummer, durch Eingabe der Art automatisch die eindeutige Artnummer abgelegt. Um nachvollziehen zu können, wer zu welchem Zeitpunkt Daten in die Datenbank gespeichert hat, sind zusätzlich Felder für die Eintragung von Anlagedatum, anlegender Person, Quellennummer, Änderungsdatum und ändernder Person vorhanden.
Die LUBW hat basierend auf der Datenbank-Anwendung FIS-Natur eine zentrale Referenzdatenbank mit dem gesamten Informationsbestand des Landes über Schutzgebiete errichtet. Ziel war es auch hier, die bedeutsamen Informationen des Naturschutzes, die bisher in den unterschiedlichsten Formaten vorlagen und über die verschiedenen Zuständigkeitsbereiche in Baden-Württemberg verteilt waren, zusammenzuführen.
Die über FIS-Natur erfassten Daten werden in den jeweiligen Dienststellen lokal gespeichert und einmal im Monat über den so genannten Datenaustauschdienst (DAD) in die Referenzdatenbank übertragen. [Lan06a]

Artenerfassungsprogramm (AEP)

Zum Zweck einer einheitlichen Datenführung wurde von der LUBW (Ref. 24, Ref. 25, ITZ) in Zusammenarbeit mit der Stadt Stuttgart, der Stadt Freiburg und dem Landkreis Ravensburg ein Programm zur Artenerfassung entwickelt, das vor allem auch im kommunalen Artenschutz seinen Einsatz findet.
Unter Berücksichtigung des Landesdatenschlüssels können mit dem AEP, bezogen auf einen bestimmten Fundort, Artenfunde und ergänzende Informationen wie Biotoptypen, Maßnah-

men, Projekte und Personen eingegeben, verwaltet und ausgewertet werden. Unter dem Landesdatenschlüssel ist eine Sammlung von Schlüssellisten zu verstehen.
Das AEP basiert auf dem Datenmodell des Fachinformationssystems Naturschutz, wurde aber um einige Kategorien erweitert, die auf eine genauere Standortsbeschreibung und auf die Umsetzung von projektbezogenen Artenschutzmaßnahmen abzielen. Wie schon bei FIS-Natur stehen auch hier der RIPS-Viewer zur Eingabe der Artenfundorte in eine Topographische Karte und der Dokumenten-Viewer zur Einbindung von Dokumenten zu einem bestimmten Fundort zur Verfügung.
Die Version 2.0 des AEP verfügt über ein Reportsystem, das die kombinierte Ausgabe sämtlicher Datenfelder mit den entsprechenden Export- und Darstellungsfunktionen ermöglicht. Bislang können die Daten nur in Form von Tabellen und Berichten ausgewertet werden. Zukünftig soll im UIS-Berichtssystem eine reduzierte und verallgemeinerte, sowie nach fachlichen Kriterien aufbereitete Funktion in Form von Selektoren und fest formatierten Reports (Berichten) bereitgestellt werden.
Momentan werden die über das AEP eingetragenen Daten noch bei jedem Nutzer in einer lokalen Access-Datenbank gespeichert. Geplant ist jedoch, dass die Daten zukünftig in das Oracle-Schema ARTIS übertragen werden. Da dieses Schema bereits in Anlehnung an die Datenstruktur des Artenerfassungsprogramms angelegt wurde, sollte die Übertragung der Daten des AEP in ARTIS mit einer geeigneten Schnittstelle problemlos möglich sein. Zuvor ist es aber nötig, die Daten aller Anwender zentral zu sammeln oder wie bei FIS-Natur einen Datenaustauschdienst anzubieten. Im AEP ist bereits ein Menü integriert, dass den Export der Funddaten ermöglichen soll. Die Version 2.0.1 des Artenerfassungsprogramms wird Ende dieses Jahres veröffentlicht. [Lan04]

Datenbank zum Artenschutzprogramm (ASP)

Unter dem Artenschutzprogramm versteht man zunächst einmal ein Instrument zur Erhaltung der Artenvielfalt in Baden-Württemberg. Dieses wird nach § 41 Abs. 1 NatSchG von der Landesanstalt für Umwelt, Messungen und Naturschutz unter Mitwirkung anderer betroffener Landesbehörden sowie der Naturschutzvereine und sachkundiger Bürger erstellt und fortgeschrieben.
Im Rahmen des ASP werden vor allem Daten zu besonders gefährdeten oder seltenen Arten erhoben. Die Entscheidung, welche Arten neu kartiert werden, trifft der Kartierer u.a. auf Basis der Grundlagenwerke, in denen Informationen über die in Baden-Württemberg vorkommenden Tier- und Pflanzenarten zusammengestellt wurden. Die erhobenen Daten erhalten dann zunächst die Regierungspräsidien, die diese noch ergänzen können. Von dort werden die Daten dann an die LUBW weitergeleitet.
Die ASP-Datenbank trägt dazu bei, die im Naturschutzgesetz beschriebenen Aufgaben im Bereich Artenschutz umzusetzen. Diese umfassen nach § 41 NatSchG:

- „*den Schutz der Tiere und Pflanzen und ihrer Lebensgemeinschaften vor Beeinträchtigungen durch den Menschen*“
- „*den Schutz, die Pflege, die Entwicklung und die Wiederherstellung der Biotope wild lebender Tier- und Pflanzenarten sowie die Gewährleistung ihrer sonstigen Lebensbedingungen*“

- „*die Ansiedlung von Tieren und Pflanzen verdrängter Arten in geeigneten Biotopen innerhalb ihres natürlichen Verbreitungsgebiets*“

Im Artenschutzprogramm sind die Schritte zur Vorbereitung, Planung und Überwachung von Maßnamen festgelegt, nicht jedoch die Maßnahmen selbst. Die in § 42 Abs. 2 des Landesnaturschutzgesetzes festgelegten Fachaufgaben des Artenschutzprogramms lassen sich grob in drei Bereiche gliedern:

1. Dokumentation der Artenbestände des Landes Baden-Württemberg
2. Vorschläge für Schutz-, Pflege- und Überwachungsmaßnahmen
3. Erfolgskontrolle mit Monitoring

[Lan06c]

Zur Vorbereitung von Maßnahmen des Biotop- und Artenschutzes gibt die Landesanstalt für Umwelt, Messungen und Naturschutz in geeigneten Zeitabständen den wissenschaftlichen Stand der Erkenntnisse über ausgestorbene und bedrohte heimische Tier- und Pflanzenarten sowie über die Gefährdung von Biotopen (Rote Listen) bekannt (§ 42 Abs. 3 NatSchG).

Die ASP-Datenbank ist eine sehr brisante Datenquelle, da sie viele Daten über vom Aussterben bedrohte oder streng geschützte Arten enthält. Deren Lebensstätten dürfen für die Öffentlichkeit keinesfalls punktgenau dargestellt werden um ihren Bestand nicht weiter zu gefährden.
Betrachtet man die Struktur der ASP-Datenbank genauer, so wird deutlich, dass die Daten einiger Spalten voraussichtlich ohne größere Schwierigkeiten in ARTIS unterzubringen sind, da dort bereits geeignete Spalten existieren. Manche Daten, wie z.B. jene, die durch die Regierungspräsidien hinzugefügt wurden, werden jedoch weniger gut zu übernehmen sein, da hierfür bislang keine Spalten im Schema ARTIS vorgesehen sind. Vor einem Import der Daten dieser Quelle in das Arteninformationssystem ist daher eine umfassende Absprache mit der Fachabteilung erforderlich!

Kartierungen von Verbänden und Artenschützern

Ehrenamtliche Kartierer, Sammler, Verbände, Natur- und Artenschützer haben in Baden-Württemberg über Jahre hinweg vielfältige Arten-Datensammlungen erstellt. Einige wichtige Daten über Artvorkommen im Land liegen bei den Naturschutzbehörden, Museen und im privaten Bereich, teils in bereits digitalisierter, teils in analoger Form vor. Um möglichst viele Artdaten in einem System bereitzustellen, sollen auch diese Informationen zukünftig in das Schema ARTIS integriert werden. Zuvor ist es aber unumgänglich, dass die Nutzungsrechte geklärt sind und vor dem Überführung der Daten eine Plausibilisierung und Kontrolle der Datenqualität stattfindet.

FLOREIN

Das DOS-Programm FLOREIN wurde zwischen 1991 und 1997 entwickelt und im Projekt „Datenbank Gefäßpflanzen“ als eigenständig laufende Datenbankanwendung auf der Basis

von DBASE (Dateien im *.dbf- Format) realisiert. Mit dem Programm können rasterbezogene Daten erfasst, bearbeitet, ausgewertet und graphisch dargestellt werden. Konzeptioniert wurde es ursprünglich zur Kartierung von Farn- und Blütenpflanzen.
Das Kartierungsraster wird normalerweise von der Topographischen Karte 1: 25.000 (nach Quadranten) abgeleitet. [Uni06]
Bedingt durch die Möglichkeit, formatfreie Datums- und Zeitangaben einzutragen, liegen diese entweder als Jahresangabe, Tag-Monat-Jahr-Angabe oder als Zeitraum mit einer von/bis-Angabe vor. Diese Eingabefreiheit ist allerdings für die Selektion der Daten nach Zeiträumen von großem Nachteil.

Biotopkartierung (§ 24a und BK 81-89)

Bereits in den siebziger Jahren erkannte man in Baden-Württemberg die Notwendigkeit, ökologisch wertvolle Bereiche im gesamten siedlungsfreien Raum zu erheben. Die Übersichtskartierung, welche die erste Phase der Kartierung darstellt, wurde von 1976 bis 1980 durchgeführt. Die zweite Phase, die Intensivkartierung, fand dann im Anschluss von 1981 bis 1989 statt. Intensiv kartiert wurden nur noch Biotope, die u.a. folgende Bewertungskriterien erfüllen:

- Naturnähe
- Vielfalt
- Seltenheit
- Gefährdung
- Vorkommende Arten (Einordnung in Rote Liste)
- Repräsentativität für eine bestimmte Biotopart
- Bedeutung für das Landschaftsbild

Die Umrisse der Biotope wurden in Karten im Maßstab 1: 25.000 eingetragen und die vorkommenden Biotoptypen, Angaben zu Wertigkeit, Gefährdung und Pflegebedürftigkeit sowie Tier- und Pflanzenarten festgehalten.
Die Daten der so genannten „Waldbiotopkartierung“, die seit 1989 unter der Leitung der Forstlichen Versuchs- und Forschungsanstalt (FVA) durchgeführt wird, sind ebenfalls im Schema ALBIS abgelegt. Die im Wald liegenden Biotope werden nicht auf Flurkarten, sondern auf vergrößerten Topographischen Karten im Maßstab 1: 10.000 eingezeichnet.
Nach Inkrafttreten des Biotopschutzgesetzes am 01.01.1992 wurde die Biotopkartierung als „§ 24a-Biotopkartierung“ weitergeführt. § 24a Abs.7 Biotopschutzgesetz verpflichtet die unteren Naturschutzbehörden, die besonders geschützten Biotope zu erheben und sie in Listen und Karten einzutragen. Um die Einheitlichkeit der landesweiten Kartierung zu gewährleisten, wurde von der LUBW eine spezielle Kartiermethode und in Zusammenarbeit mit der Firma Vedewa ein Erfassungsprogramm entwickelt. Die in den Wäldern gelegenen Biotope wurden in Absprache mit dem Umweltministerium und dem Ministerium Ländlicher Raum von der FVA erfasst.

Heute ist die Biotopkartierung ein wichtiger Baustein des UIS. Die Ergebnisse dieser Kartierung stehen Behörden und freien Planern zur Verfügung. Seit der Novellierung des Naturschutzgesetzes im Dez. 2005 werden die besonders geschützten Biotope nun im § 32 NatSchG aufgeführt. Ende 2004 war der erste Durchgang der Kartierung landesweit abgeschlossen. Die erhobenen Daten werden im Oracle Datenbankschema ALBIS an der LUBW geführt, das Ende 2006 auf dem aktuellen Stand „konserviert“ wird. Die Daten der § 24a- (jetzt § 32-) Kartierung werden dann in das derzeit noch im Aufbau befindliche Datenbankschema NAIS überführt. [Lan95]

Zum beispielhaften Import von Daten in ARTIS über das zu entwickelnde Konvertierungswerkzeug wurde die Datenquelle der § 24a-Biotopkartierung ausgewählt, da es sich bei dieser Quelle um den größten systematischen Artdatenbestand Baden-Württembergs handelt und diese Daten bereits im Oracle Schema ALBIS vorlagen. Weitere Details zum Schema ALBIS finden sich in Kapitel 2.3. An dieser Stelle sei darauf hingewiesen, dass in der vorliegenden Arbeit die Daten dieser Kartierung mit „B24a“ bzw. § 24a bezeichnet werden, obwohl diese Daten in der neuesten Version des „Gesetzes zum Schutz der Natur, zur Pflege der Landschaft und über die Erholungsvorsorge in der freien Landschaft“(Naturschutzgesetz - NatSchG, Stand: Dezember 2005) im § 32 aufgeführt sind. Der Grund für die Verwendung der „alten“ Paragraphenbezeichnung im Text liegt darin, dass die meisten Namen der Tabellen des Datenbankschemas ALBIS die Bezeichnung „24a“ enthalten. Um den Leser nicht unnötig durch die Verwendung der neuen Paragraphenbezeichnung zu verwirren bzw. fortlaufend „§ 24a (jetzt § 32)“ schreiben zu müssen, wird in der ganzen Diplomarbeit die Bezeichnung „24a“ benutzt.

2.2.3 Vorteile eines zentralisierten ARTIS

Die Vorteile eines gemeinsamen Artdatenbestands sind vielfältig. Nachfolgend sollen die wichtigsten kurz erläutert werden.

1. **Vereinfachung und Beschleunigung der Auswertung**
   Durch die Zusammenfassung aller Artdaten in einem System erübrigt sich die Suche nach einer bestimmten Art in verschiedenen Datenquellen. So ließe sich beispielsweise die Auswertung und Zusammenstellung bestimmter Daten über die Artvorkommen Baden-Württembergs im Rahmen der Berichtspflichten gegenüber der EU vereinfachen und erheblich beschleunigen.

2. **Verbesserung des Informationsaustauschs über die Artvorkommen in BW**
   Greifen alle mit dem Artenschutz befassten Stellen auf den gleichen Artdatenbestand zurück, wird die Zusammenarbeit dieser Stellen wesentlich verbessert.

3. **Korrektur und bessere Verwaltung der Artdaten**
   Ein gemeinsamer Artdatenbestand in einer zentralen Datenbank ist einfacher zu verwalten und zu warten als verschiedene heterogene Datenbestände. Zudem lässt sich der Prozess der Qualitätssicherung erheblich vereinfachen.

4. **Optimierung der Nutzung der Daten zu Tier- und Pflanzenarten**
   Liegt zukünftig ein plausibilisierter Datenbestand zu Tier -und Pflanzenarten vor, können Teile dieser wichtigen Daten des Artenschutzes über verschiedene Anwendungen auch der Öffentlichkeit zugänglich gemacht werden.

2.2.4 Technische und fachliche Probleme bei der Integration der verschiedenen Datenbestände in ARTIS

Die Daten der in Kapitel 2.2.2 beschriebenen Datenquellen liegen in den verschiedensten Systemen und Formaten mit unterschiedlichen Strukturen vor. Eine genaue Analyse der Datenstrukturen ist daher vor jedem Import einer Datenquelle notwendig. Dieser arbeits- und zeitaufwendige Schritt ist unvermeidbar, da zu jeder Datenquelle eine eigene Schnittstelle entwickelt werden muss, um alle artrelevanten Informationen in das Zielschema überführen zu können. Eine Schnittstelle, die für Importe aus allen erdenklichen Datenquellen verwendet werden kann, ist aufgrund der vielfältigen Strukturen nicht umsetzbar.
Neben dem unterschiedlichen Aufbau der Datenquellen können auch die voneinander abweichenden Datentypen ein Problem darstellen. Auf diesen Punkt und andere mögliche Importprobleme wird an weiteren Stellen dieser Arbeit noch genauer eingegangen.

Die strukturelle Analyse geht einher mit der inhaltlichen Analyse der Datenquellen. Weite Teile dieser inhaltlichen Analyse, zu der auch die Plausibilisierung der Daten gehört, können nur von Fachleuten vorgenommen werden. Dieser Schritt ist von großer Bedeutung für die Sicherung der Datenqualität und zur Schaffung einer guten und verlässlichen Basis für zukünftige Auswertungen.
Die im Rahmen der Diplomarbeit importierten Daten der § 24a-Biotopkartierung konnten aus internen Gründen bisher nicht von der Fachabteilung plausibilisiert werden. Dennoch sollten die Daten importiert werden, da der entstehende Kartendienst, mit dessen Aufbau sich der zweite Teil der Diplomarbeit beschäftigt, auf die Daten des Schemas ARTIS inklusive des Datenbestands der § 24a-Biotopkartierung zugreifen soll. Zudem kann der Kartendienst in bestimmten Grenzen auch als Werkzeug zur Plausibilisierung der Daten dienen. So wäre es z.B. denkbar, dass ein Experte die Verteilung bestimmter Arten aus einer oder mehreren Artgruppen im Kartendienst betrachtet und so die Plausibilität der Artenfunde beurteilen und bewerten kann.
Durch die Hervorhebung der Artenfundorte in der Karte von Baden-Württemberg ist die Verbreitung einer Art erkennbar. Anhand dieser Darstellung und dem Vergleich mit anderen Quellen kann der Experte erkennen, ob die ihm präsentierten Funde als plausibel einzustufen sind.
Neben der allgemeinen Plausibilisierung der Datenbestände müssen von der Fachseite auch Toleranzgrenzen zur Vermeidung von Duplikaten bei der Zusammenführung aus den verschiedenen Datenbeständen definiert werden. Dazu gehört zum Beispiel, dass für jede Artengruppe festgelegt werden muss, in welchem Radius Artenfunde noch als ein Fund oder eben als verschiedene Artenfunde zu betrachten sind. So würde man Artenfunde im gleichen Radius bei kleinräumig verbreiteten Pflanzengruppen eher als getrennte Funde betrachten als beispielsweise bei Tiergruppen mit ausgedehnten Revieren. Bei dieser Entscheidung muss auch der Aspekt der Kartiergenauigkeit berücksichtigt werden.
Sind alle zu beachtenden fachlichen Gesichtspunkte definiert, kann versucht werden, programmtechnisch eine Filterung der als nicht plausibel eingestuften Daten vorzunehmen. Ob dies sinnvoll und möglich ist, wird erst der Praxistest beweisen. Da diese Entwicklung jedoch den Umfang der Diplomarbeit sprengen würde und die Plausibilisierung der Daten zuvor noch viel Zeit in Anspruch nehmen wird, soll sie hier unbeachtet bleiben.

2.3 Das Datenbankschema ALBIS

In ALBIS werden die dezentral erfassten Daten der landesweit betriebenen Biotopkartierung nach § 24a NatSchG, die Naturschutz-Datenschlüssel Baden-Württemberg (Landesdatenschlüssel) sowie die Daten aus dem Artenlexikon des Landes gehalten.
Bis jetzt werden diese u.a. vom UIS-Berichtssystem genutzt. Neben der Abfrage von Biotopen ist dort auch eine Auswertung der in Baden-Württemberg lebenden Arten möglich. Es soll hier kurz beschrieben werden, wo diese Möglichkeit innerhalb des Berichtssytems zu finden ist. Über die vom Berichtssystem erzeugte Log-Datei kann der Ablauf einer Abfrage nachverfolgt werden. Außerdem ist dort erkennbar auf welche Tabellen des ALBIS Schemas bei einer Abfrage zugegriffen wird.
Wählt man über den Themenbaum des Berichtssystems über – Referenzdaten→Sachdaten→ Naturschutz, Landschaftsplanung→Flächenschutz, Landschaftspflege→Biotope nach NatSchG und LWaldG→R Biotope – die Auswertung nach Arten aus, kann eine Abfrage nach einer bestimmten Art gemacht werden. Wird diese gestartet, baut sich eine Artenliste auf, die die Kurzbezeichnung, den deutschen Namen und den wissenschaftlichen Namen enthält. Dabei greift das UIS-Berichtssystem im Hintergrund auf die Tabellen zu, die die benötigten Daten enthalten. Es handelt sich um die ALBIS-Tabellen AL_ART, AL_DNAME und B24AV_ZUORD_BIO_ARTEN. Diese Tabellen werden später auch für den Kartendienst bzw. den Einstieg in den Kartendienst eine Rolle spielen.
Wählt man nun ein oder mehrere Arten aus, werden diese in der Geokomponente „GISterm“ auf einer Hintergrundkarte als „Punktdarstellung des Biotops“ ausgegeben. Lädt man sich dann die Biotopflächen über den Themenbrowser hinzu, kann man erkennen, dass der Artenfundort als Mittelpunkt der Biotopfläche verstanden werden muss (siehe Abbildung 2.2).
Neben dem UIS-Berichtssystem nutzen auch die Fachanwendungen des Naturschutzinformationssystems (NAIS) ALBIS als Referenzdatenquelle.

2.3.1 Analyse des Schemas ALBIS

Die Analyse dieses Datenbankschemas diente vor allem dazu herauszufinden, in welchen Tabellen und Spalten Daten zu Arten und Artenfunden gehalten werden. Für die Arbeit mit den Oracle Datenbankschemata wird in der LUBW die Datenbanksoftware Toad™ for Oracle verwendet. Dieses Werkzeug wurde auch zur Analyse des Datenbankschemas ALBIS verwendet. Aus diesem Grund soll in Kapitel 2.3.2 ein kurzer Überblick über die wichtigsten Funktionaliäten dieser Software gegeben werden.

Da es im Rahmen der Diplomarbeit zunächst nur um den Import von Daten der § 24a-Biotopkartierung geht, wurde die Analyse zunächst auf die Tabellen beschränkt, die das Kürzel „B24a“ im Namen enthalten. So mussten insgesamt 54 Tabellen genauer betrachtet werden. Im nächsten Schritt wurde dann anhand der Spaltennamen dieser Tabellen versucht herauszufinden, welche Spalten Informationen zu Artenfunden enthalten. War dies dem Namen nicht eindeutig zu entnehmen, wurde die Rubrik „Comments“, die Erläuterungen zu den einzelnen Spalten enthält, zu Hilfe genommen. Leider ließ sich so nicht immer eindeutig bestimmen, ob Artdaten in der betrachteten Spalte abgelegt sind. Daher war es im nächsten Schritt nötig, sich die Daten genauer anzusehen. Anhand der Spalteneinträge war dann meist erkennbar, ob es sich um Daten handelt, die für den Import in ARTIS relevant sind oder nicht. Letzte Unklarheiten konnten durch Anfrage bei den Datenbankexperten des ITZ bzw.

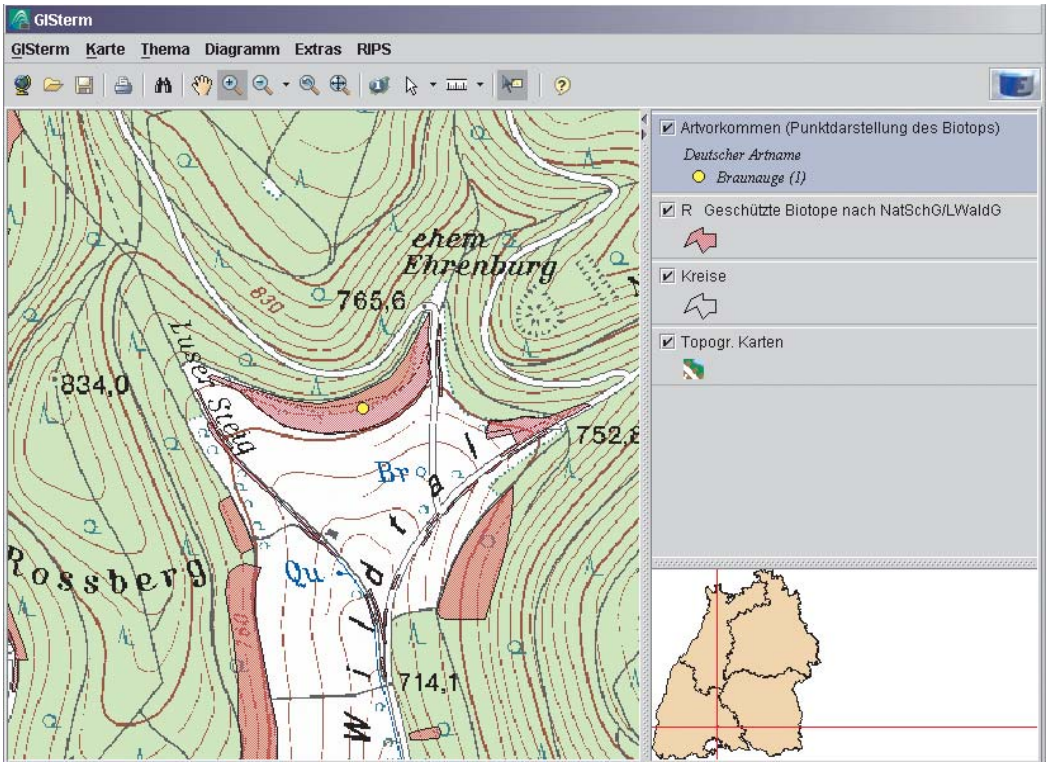


Abbildung 2.2: Darstellung von Artenfundort und Biotopfläche in GISterm

der Fachabteilung geklärt werden. Die Analyse kam zu folgendem Ergebnis:

Informationen zu Artenfunden befinden sich in den Tabellen:

- B24A_AB_BIOTOP
- B24A_BIOTOP
- B24A_PERSON

Zur besseren Unterscheidung von Tabellen und Spaltenbezeichnungen werden sowohl in diesem Kapitel wie auch Kapitel 3 und 4 die Tabellennamen grundsätzlich komplett in Großbuchstaben (z.B. AE_PERSON) geschrieben, wohingegen die Spaltenbezeichnungen (z.B. *p_nname*) komplett klein und kursiv geschrieben werden.
Die zuvor genannten Tabellen sollen im folgenden Unterkapitel zur genaueren Analyse einander gegenübergestellt werden.
Zuvor jedoch noch ein paar statistische Werte: Das Schema ALBIS wird, wie bereits erwähnt, zum Ende dieses Jahres auf dem aktuellen Stand (15.11.06) „eingefroren“. Damit enthält es ca. 2 000 000 Datensätze, in denen Daten zu ca. 6000 verschiedenen Tier- und Pflanzenarten, die an ca. 78 000 unterschiedlichen Fundorten kartiert wurden, gespeichert sind.

2.3.2 Das Datenbankwerkzeug Toad for Oracle

Toad^{TM} for Oracle ist das mächtige Datenbank-Entwicklungs- und -Verwaltungswerkzeug der Firma Quest Software, Inc., das den Umgang mit Datenbanken erheblich erleichtert. So bietet die Software beispielsweise die Möglichkeit, mehrere Schemata nebeneinander zu öffnen, was den Vergleich der Strukturen wesentlich vereinfacht. Öffnet man eine bestimmte Tabelle, so erhält man eine übersichtliche Liste mit den Spaltennamen, der Angabe bei welchem Feld bzw. bei zusammengesetzten Schlüsseln bei welchen Feldern es sich um den Primärschlüssel handelt, der Angabe, ob das Feld den Wert NULL annehmen darf, den Datentyp und Kommentare zu der jeweiligen Spalte. Gerade die Kommentare sind bei der Analyse von entscheidender Bedeutung, da die Feldnamen oftmals leider nicht selbsterklärend sind.

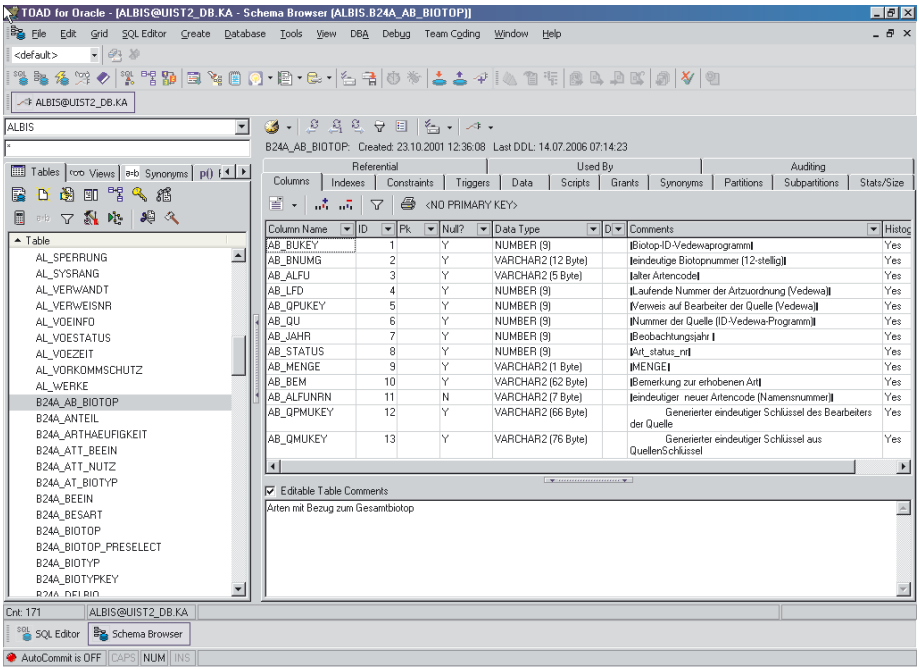


Abbildung 2.3: Benutzeroberfläche der Software Toad^{TM} for Oracle

Zum Verständnis der Beziehungen zwischen den einzelnen Tabellen werden im Reiter „Referential“ die Tabellen aufgelistet, zu denen die vorliegende Tabelle in einer bestimmten Beziehung steht. Über einen integrierten SQL-Editor können Abfragen abgesetzt werden, Tabellen und Sichten neu angelegt, verändert, gelöscht oder Zugriffsrechte vergeben werden. Das Ergebnis einer Abfrage erscheint dann in einem Fenster unter dem Codefenster. Somit bleibt der Nutzer von allzu vielen Wechseln zwischen den Fenstern verschont. Die anwenderfreundliche grafische Benutzeroberfläche (siehe Abbildung 2.3) lässt sich nutzerspezifisch anpassen und vereinfacht so die Anwendung der Software, auch für den im Umgang mit Datenbanken wenig erfahrenen Nutzer. Für den Experten gibt es eine integrierte Entwicklungsumgebung, die u.a. das Anlegen von Schemata und die Ausführung von SQL-Skripten ermöglicht. Die „Toad Community“ stellt zudem allen Anwendern ein Forum zur Kommunikation und zum Erfahrungsaustausch zur Verfügung.

2.3.3 Gegenüberstellung der Tabellen der ALBIS und ARTIS Schemata

Nachdem die für den Import benötigten Tabellen sowohl im Quellschema ALBIS wie auch im Zielschema ARTIS ausfindig gemacht worden waren, wurden die Tabellen einander gegenübergestellt. Ziel war es aufzeigen, aus welcher Spalte der ALBIS-Tabellen Daten in die entsprechende Spalte der ARTIS-Tabellen importiert werden können. Dabei musste besonders darauf geachtet werden, ob sich die Datentypen der zugeordneten Spalten unterscheiden, da dies zu Problemen beim Import führen kann. Auf der linken Seite der Abbildungen 2.4, 2.5 und 2.6 sieht man jeweils die Tabelle des Quellschemas ALBIS, aus der die Daten der umrandeten Spalten in ARTIS übernommen werden sollen. Über die Pfeile lässt sich erkennen, in welche Spalten der ARTIS-Tabelle die Daten importiert werden. Um die Lesbarkeit zu erhöhen, wurden unterschiedliche Pfeilfarben gewählt.

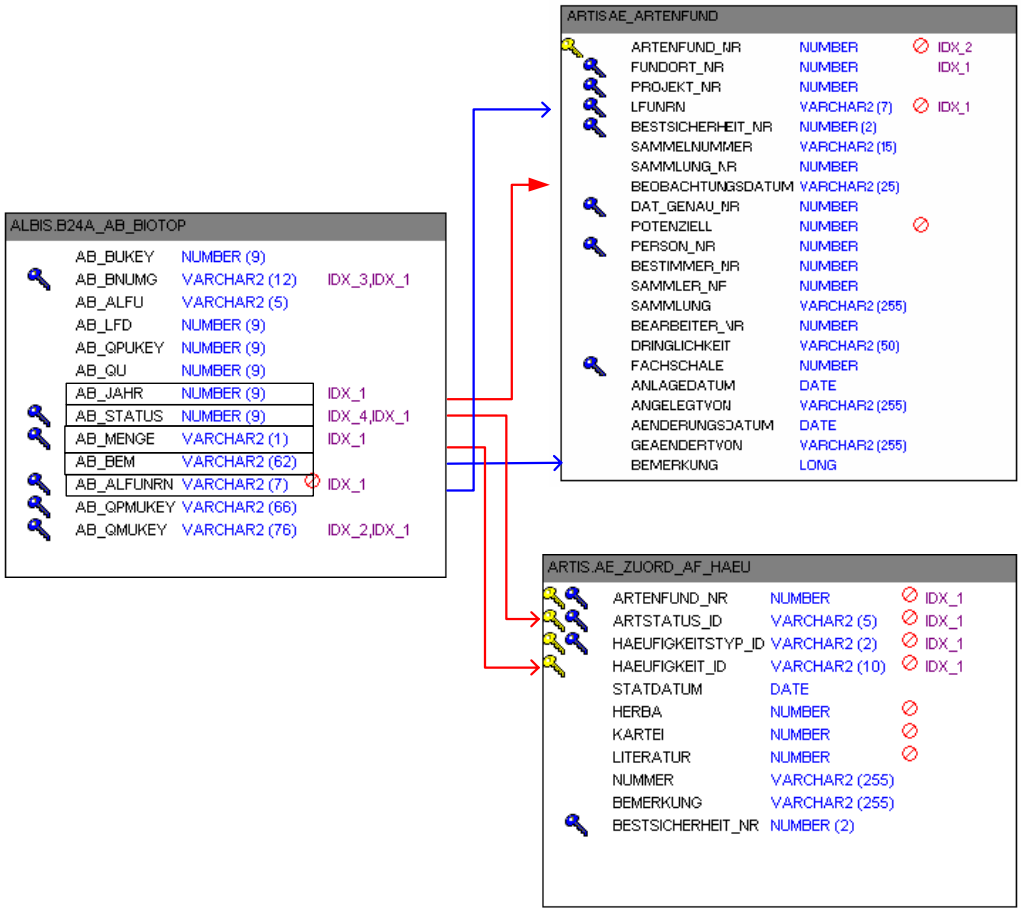


Abbildung 2.4: Gegenüberstellung der Tabelle ALBIS.B24A_AB_BIOTOP mit den Tabellen ARTIS.AE_ARTENFUND und ARTIS.AE_ZUORD_AF_HAEU

Die Tabelle ALBIS.B24A_AB_BIOTOP ist die Tabelle, welche die meisten Artinformationen enthält. In der Spalte *ab_jahr* ist das Beobachtungsdatum der Art, in a*b_status* der Status und in *ab_menge* die Häufigkeit des Vorkommens der Art abgelegt. Die Spalte *ab_bemerkung* enthält ergänzende Informationen zur erhobenen Art. In *ab_alfunrn* ist die Artnummer gespeichert, über die eine Art eindeutig identifizierbar ist.

Beim Vergleich dieser Tabellen wird deutlich, dass sich die Datentypen der Quell- und Zielspalte häufig unterscheiden. Importe von Daten aus Spalten vom Datentyp `varchar2` in Spalten vom Typ `number` oder vom Typ `date` sind immer ohne Probleme möglich, sofern es sich beim Eintrag eines Feldes vom Typ `varchar2` um eine gültige Nummerndarstellung handelt. In diesem Fall wird die Konversion implizit vom Server übernommen. Auch umgekehrt ist keine explizite Konvertierung notwendig. Ebenfalls ohne Probleme können Daten aus einem Feld vom Typ `varchar2` in ein Feld vom Typ `long` übernommen werden, da es sich bei beiden Datentypen um Zeichenketten variabler Länge handelt.
Die Artenfundtabelle von ARTIS (siehe Abbildung 2.4, rechte Seite) speichert alle Informationen über eine Art, außer dem Artstatus und der *haeufigkeit_id*. Diese Informationen werden in der Tabelle ARTIS.AE_ZUORD_AF_HAEU abgelegt. Wie die Informationen über den Artstatus abgelegt sind, kann Tabelle 4.1 entnommen werden.
In die Spalte *anlagedatum* der Tabelle ARTIS.AE_ARTENFUND werden die Daten aus der Spalte *bio_erfdat* (=Erfassungsdatum) der Tabelle ALBIS.B24A_BIOTOP importiert (siehe Abbildung 2.5). Genau diese Datumsangabe wird auch nochmals in die Tabelle ARTIS.AE_FUNDORT geschrieben, auf die in Kapitel 4.1.6 noch genauer eingegangen wird. Der Grund für diese Eintragungsweise ist, dass in der Tabelle B24A_AB_BIOTOP für eine Art kein Anlagedatum gespeichert ist. Da man aber davon ausgehen kann, dass Art und Biotop zum gleichen Zeitpunkt in der Datenbank angelegt worden sind, ist diese Speicherungsweise möglich.

Die Tabelle ALBIS.B24A_BIOTOP speichert die Informationen über Biotope. Diese enthält auch den Fundort, der durch die Koordinaten des Biotopmittelpunkts beschrieben. Daher werden aus dieser Tabelle die Rechts- und Hochwerte, die Nummer des Blattes der Topographischen Karte 1: 25.000 (TK 25), in dem das Biotop liegt, der Biotopname, das Bearbeitungsdatum und wie bereits erwähnt, das Erfassungsdatum übernommen.
Da die Spaltennamen der Tabelle ARTIS.AE_FUNDORT selbsterklärend sind, soll in diesem Fall nicht näher auf sie eingegangen werden. In welche Spalten dieser Tabelle die Daten aus der Quelltabelle importiert werden, lässt sich Abbildung 2.5 entnehmen. Die Daten, die in die Spalten *tk_quadrant* und *geologie_id* importiert werden sollen, werden den Tabellen B24A_ZUORD_BIO_TK bzw. B24A_IGEOBIO und B24A_GEOLOGIE entnommen. Da jeweils nur Werte einer Spalte dieser Tabellen importiert werden und um die Übersichtlichkeit der Grafik zu wahren, wurden diese Tabellen nicht mit in die Abbildung aufgenommen. In der Grafik wurden diese Spalten mit einem Sternchen gekennzeichnet. Ein Entity-Relationship-Diagramm (ER-Diagramm) des Schemas ALBIS, auf dem alle Tabellen der § 24a-Biotopkartierung und weitere mit ihr in Verbindung stehende Tabellen dargestellt sind, befindet sich in Anhang D.

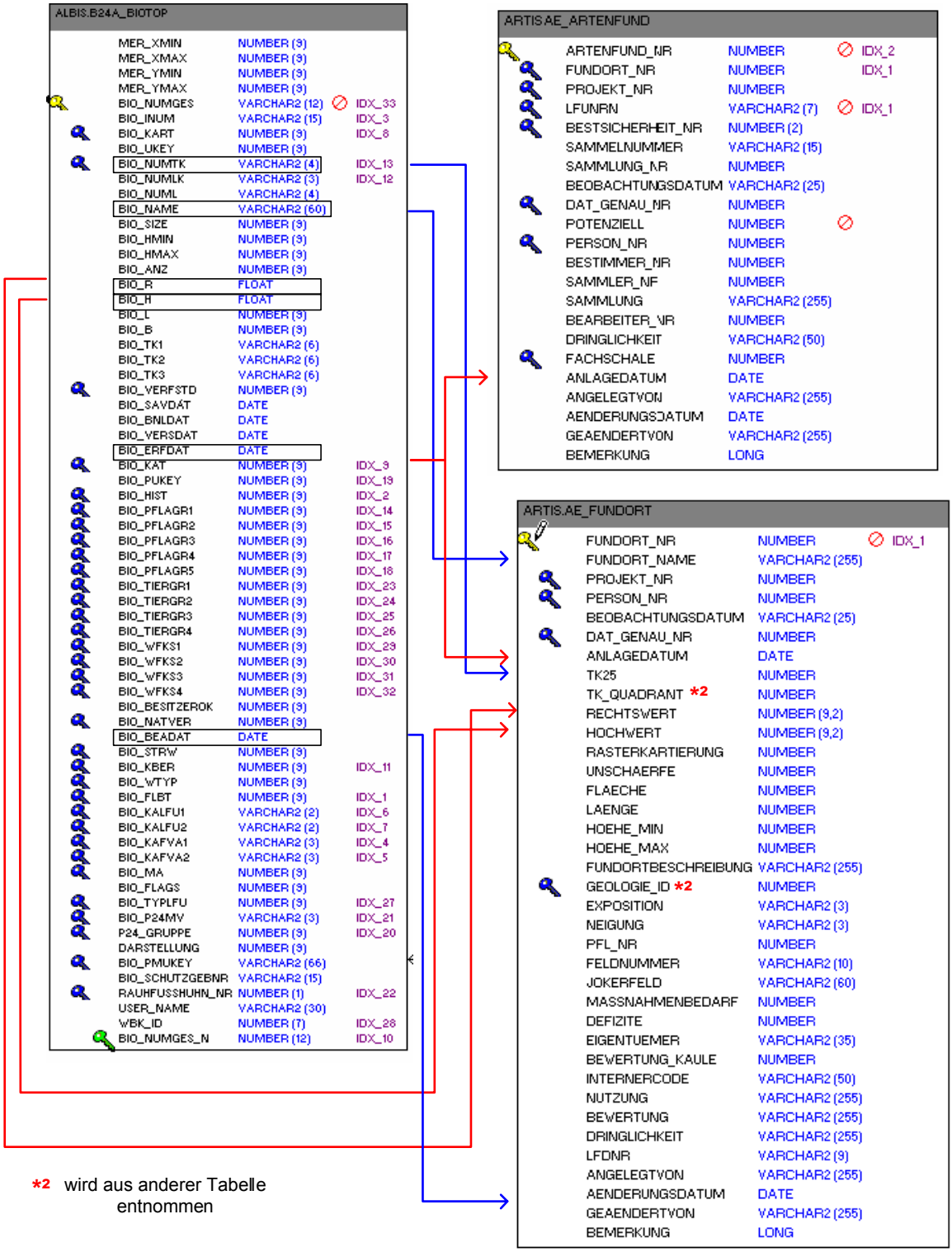


Abbildung 2.5: Gegenüberstellung der Tabelle ALBIS.B24A_BIOTOP mit den Tabellen ARTIS.AE_ARTENFUND und ARTIS.AE_FUNDORT

Neben den Daten zu Biotopen und Artenfunden müssen auch Personendaten mit in ARTIS übernommen werden. Es handelt sich hierbei um die Adressdaten der Kartierer. Aus Datenschutzgründen dürfen diese Daten selbstverständlich nicht in einem der Öffentlichkeit zugänglichen Kartendienst ausgegeben werden. Dennoch spielt die Angabe des Kartierers gerade für die Plausibilisierung der Daten durch die Fachleute eine große Rolle.
Abbildung 2.6 zeigt die Gegenüberstellung der Tabellen ALBIS.B24A_PERSON und ARTIS.AE_PERSON. Vorwahl und Telefon- bzw. Faxnummer, die in ALBIS in zwei verschiedenen Spalten abgelegt waren, werden über eine SQL-Funktion verkettet und in die Spalten *telefon* bzw. *fax* der Zieltabelle eingetragen. Die Einträge der Spalte *p_wohnort* müssen über eine View entschlüsselt werden, da der Wohnort in der Quelltabelle in einer Ziffernfolge codiert abgelegt wurde. Dies kann man bereits am Datentyp `number` erkennen. Eine „View“ ist eine Sicht auf die Daten und wird auch als virtuelle Tabelle bezeichnet. Diese dient in erster Linie der Vereinfachung der SQL-Abfragen, da die View wie eine normale Tabelle abgefragt werden kann und daher keine komplexen Abfragen über mehrere Tabellen notwendig sind. Bei jedem Zugriff auf die View wird diese durch das Datenbankmanagementsystem neu berechnet, so dass sie immer die neuesten Daten der Tabellen, auf die sie zugreift, enthält.
Eine weitere Besonderheit der Personentabelle ist die Spalte *p_anrede*. Sie ist vom Datentyp `number` und enthält die Werte 0 für keine Anrede, 1 für Herr und 2 für Frau. Diese müssen vor dem Eintrag in ARTIS über eine View entschlüsselt werden.

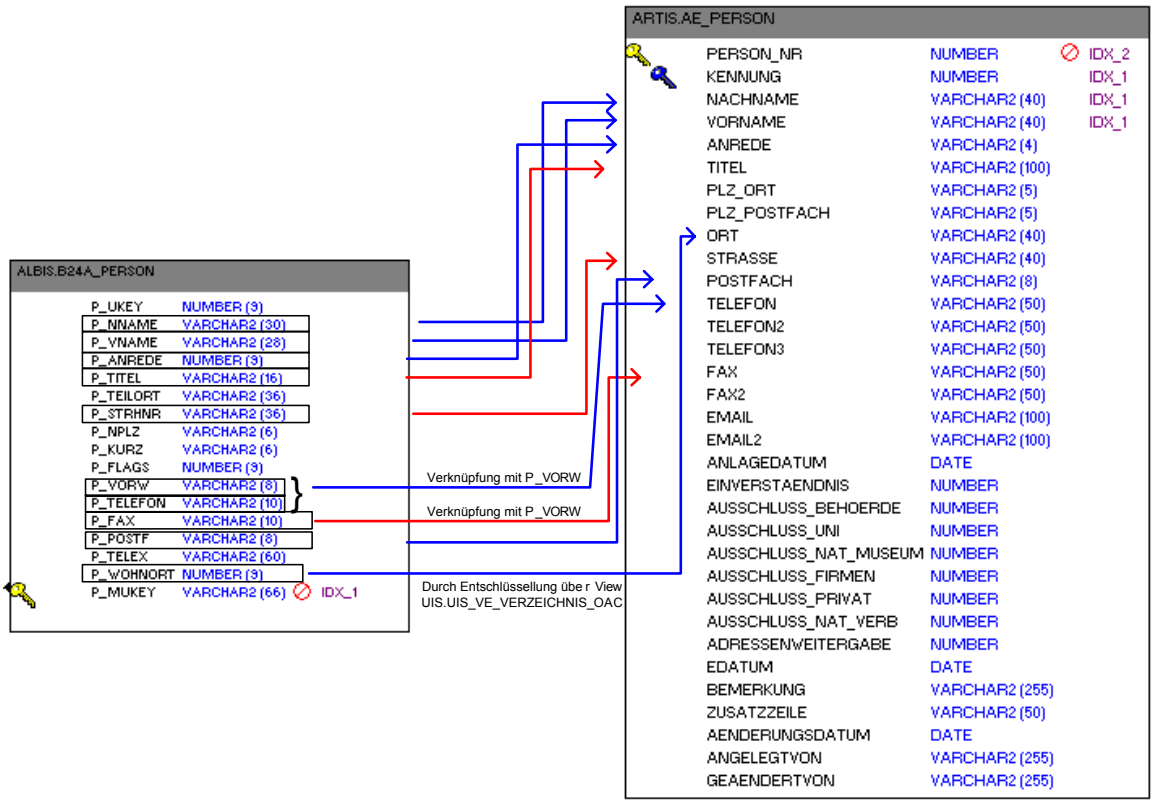


Abbildung 2.6: Gegenüberstellung der Personentabellen von ALBIS und ARTIS

3 Erstellung eines Konzepts zur Datenübernahme

Zu Beginn jedes Projekts muss festgelegt werden, welche Arbeitsschritte zur Erreichung des Ziels notwendig sind. Das hier vorliegende Projekt erfordert neben der Analyse der Ausgangssituation und der für den Import in ARTIS vorgesehenen Datenquellen auch eine Zusammenstellung der Anforderungen an das Konvertierungswerkzeug sowie eine Evaluierung der technischen Möglichkeiten zur Realisierung der Datenübernahme.

3.1 Anforderungen an das Konvertierungswerkzeug

Zunächst müssen die fachlichen und technischen Anforderungen, die beim Import in das Schema ARTIS beachtet werden sollen, zusammengestellt werden. Diese werden nachfolgend aufgelistet:

1. Einmal pro Importvorgang soll eine eindeutige Nummer (Import_ID) vergeben werden. Als Importvorgang wird der Ablauf von Programmstart bis Programmende gesehen.

2. Es soll nur eine neue Personennummer (*Person_Nr*) erzeugt und in die Tabelle ARTIS.AE_PERSON eingetragen werden, wenn die Person in der Zieltabelle noch nicht existiert. Dies dient der Vermeidung redundanter Einträge.

3. Eine neue Fundortnummer soll nur erzeugt werden, wenn der Fundort in der Zieltabelle noch nicht vorkommt. Ist dies der Fall, soll innerhalb eines Importvorgangs nur dann eine neue Fundortnummer erstellt werden, wenn sich entweder die Biotopnummer ändert oder bei gleicher Biotopnummer eine andere Personennummer auftritt. Wurde das Biotop also von zwei verschiedenen Personen kartiert, so muss es auch zweimal in die Zieltabelle aufgenommen werden, da diese Information sonst verloren ginge.

4. Zu jeder Art soll der Artstatus, der zusätzliche Informationen zur „Lebenssituation“ der Art enthält, gespeichert werden. (siehe auch Kapitel 4.1.8)

5. Ein Artenfund soll an einem Fundort nur einmal eingetragen werden, außer, er wurde in verschiedenen Jahren dort beobachtet. Dann soll diese Information erhalten bleiben.

6. Nach dem Eintrag eines Datensatzes in eine Zieltabelle muss vom Programm in die Importtabelle geschrieben werden, welcher Datensatz importiert wurde. Der Eintrag soll eindeutig anhand der Kombination aus Importnummer (import_id), Tabellennummer (tab_id), eindeutiger Nummer aus der Quelltabelle (alt_nr) und eindeutiger Nummer aus der Zieltabelle (neu_nr) identifizierbar sein.

7. Generell muss darauf geachtet werden, ob die Spalten der Zieltabelle einen „NOT NULL-Constraint“ besitzen. Ein „NOT NULL-Constraint“ verhindert, dass eine Spalte leer

bleibt, in die zwingend Werte eingetragen werden müssen. Ist in der Quelltabelle im betreffenden Feld kein Wert vorhanden, so muss in das Feld der Zieltabelle die Zahl 0 eingetragen werden.

8. Alle Importvorgänge und dabei eventuell aufgetretene Fehler sollen in einer Log-Datei protokolliert werden.

3.2 Werkzeuge zur Datenübernahme

Zur Übernahme von Daten aus einer Datenquelle in eine andere bieten sich verschiedene Möglichkeiten an. Grundsätzlich muss die Entscheidung getroffen werden, ob ein bestehendes Softwareprodukt verwendet werden soll oder ob das Werkzeug eigens programmiert werden soll. Beide Möglichkeiten haben Vor- und Nachteile, auf die in Kapitel 3.3 näher eingegangen wird. Als Beispiel für ein Werkzeug zur eigenen Programmierung einer Schnittstelle soll Visual Basic in der Entwicklungsumgebung Visual Studio 6.0 genauer betrachtet werden, da es die Möglichkeit bietet, auf relativ einfache Weise ein Programm mit einer grafischen Benutzeroberfläche zu erstellen. Als Beispiel für eine bestehende Software soll der Oracle Warehouse Builder 10.2.0.1 vorgestellt werden.

3.2.1 Visual Basic Programmierung mit Microsoft Visual Studio 6.0

Für die Programmierung eines Konvertierungswerkzeugs zur Übernahme der Artdaten aus dem Schema ALBIS in das Schema ARTIS kann das Entwicklungswerkzeug Microsoft Visual Studio 6.0 verwendet werden. Hierbei handelt es sich um eine komfortable Entwicklungsumgebung, die sich u.a. zur Programmierung von Anwendungen in Visual Basic, C++ und J++ eignet. Neben dem Modul zur Programmierung in Visual Basic (VB) sind noch folgende weitere Module enthalten:

* Visual InterDev – eine Anwendung zur Entwicklung von Webseiten, die zur Modifizierung von Active Server Pages, HTML-Seiten und anderen Web Scripting Files verwendet werden kann
* Visual C++ – eine Entwicklungsumgebung zur Programmierung in C++
* Visual J++ – eine Java-Entwicklungsumgebung
* Visual FoxPro – eine für daten-zentrierte Anwendungen entwickelte Rapid Application Development-Umgebung [Wik06a]

Zum Einstieg in die Programmierung mit VB soll zunächst eine kurzer Überblick über die Sprache Visual Basic gegeben werden.
Bei Visual Basic handelt es sich um eine *ereignisorientierte* Programmiersprache, die auf der Programmiersprache BASIC (Beginners All-purpose Symbolic Instruction Code) basiert. Die Sprache wird als *ereignisorientiert* bezeichnet, da der Programmablauf durch Handlungen des Benutzers, z.B. durch Drücken einer Schaltfläche oder die Folge von Befehlen, gesteuert werden kann. In der Fachliteratur wird die Sprache manchmal auch als *objektorientiert* bezeichnet, was jedoch nicht ganz korrekt ist, da die Sprache zwar seit der Version 4 objektorientierte Elemente enthält, jedoch nicht alle Kriterien einer objektorientierten Programmiersprache erfüllt [Wik06b].

Wie in Abbildung 3.1 dargestellt, wird in Visual Studio 6.0 zur Erstellung einer Visual Basic Anwendung zunächst ein neues Projekt angelegt. Dieses kann aus einem oder mehreren Modulen bestehen. Als Modul wird dabei sowohl ein Standardmodul, in dem der allgemeine Programmcode steht, als auch ein Formularmodul, das, wie der Name schon sagt, ein Formular enthält, bezeichnet. Über das Formularmodul wird die Benutzeroberfläche definiert. Jedes Modul enthält sogenannte Prozeduren. Prozeduren sind Programmabschnitte, die Code enthalten und das Programm in kleinere logische Einheiten aufteilen. Drei Arten von Prozeduren lassen sich unterscheiden:

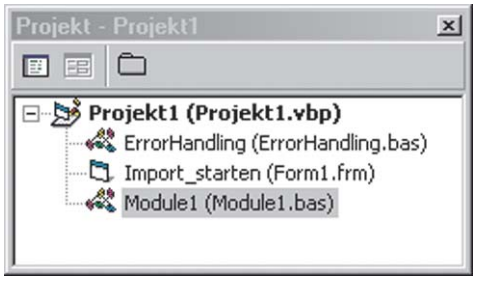


Abbildung 3.1: Aufbau einer VB-Anwendung

1. Sub-Prozeduren: Besitzen keinen Rückgabewert
2. Function-Prozeduren: Besitzen einen Rückgabewert
3. Property-Prozeduren: Können Werte zurückgeben, zuweisen und auf Objekte verweisen

Function-Prozeduren werden zur Vereinfachung im Folgenden immer als *Funktionen* bezeichnet. Prozeduren können sowohl über den Prozedurnamen wie auch mit dem Schlüsselwort *Call* aufgerufen werden. Prozeduren , die mit dem Schlüsselwort *Private* bezeichnet werden, sind nur in dem Modul gültig, in dem sie deklariert wurden, wohingegen mit *Public* bezeichnete Prozeduren im ganzen Projekt aufrufbar sind.
Eine weitere Besonderheit von VB stellt die Variablendefinition dar. Die einfachste Methode eine Variable zu deklarieren, ist über das Schlüsselwort *DIM*, gefolgt vom Namen der Variablen. Grundsätzlich ist es in Visual Basic möglich, Variablen ohne Angabe des Datentyps zu deklarieren. Es ist jedoch empfehlenswert, die Variable stets unter Angabe des Datentyps zu deklarieren, da dieser dem Programm sagt, wofür die Variable im Programm verwendet werden soll und damit, wie viel Speicherplatz vom System für den Inhalt der Variablen reserviert werden muss. Um den Programmierer zur Angabe des Datentyps zu zwingen, muss die Option *Explicit* zu Beginn des Programmcodes angegeben werden. Eine Variablendeklaration sieht dann wie folgt aus:

Dim *Variablenname* **As** *Datentyp*

Da das Konvertierungswerkzeug hier zum Import von Daten aus einem Oracle Datenbankschema in ein anderes dienen soll, wird eine Schnittstelle benötigt, die den Zugriff auf Datenbanken erlaubt. Hier bietet sich die ActiveX Data Objects (ADO)-Schnittstelle an, über die die meisten gebräuchlichen Datenbanken angesprochen werden können.

Das ADO-Objektmodell

ADO basiert auf ActiveX, einer von Microsoft entwickelten Technologie, die es ermöglicht, interaktive Inhalte auf einer Webseite darzustellen. ADO ist nichts anderes als ein Objektmodell zur einfachen Handhabung der ODBC-Schnittstelle (Open DataBase Connectivity).

Dieses wird neben einem OLE-DB-Provider (Object Linking and Embedding Database) für Oracle für den Zugriff auf eine Datenbank benötigt. Über ADO kann die Verbindung zu einer Datenbank hergestellt werden, ohne die Datenbankabfragesprache SQL verwenden zu müssen. Es besteht jedoch die Möglichkeit, SQL-Abfragen in den Visual Basic Programmcode einzubinden.

Abbildung 3.2 zeigt die Hierarchie der ADO-Objekte:

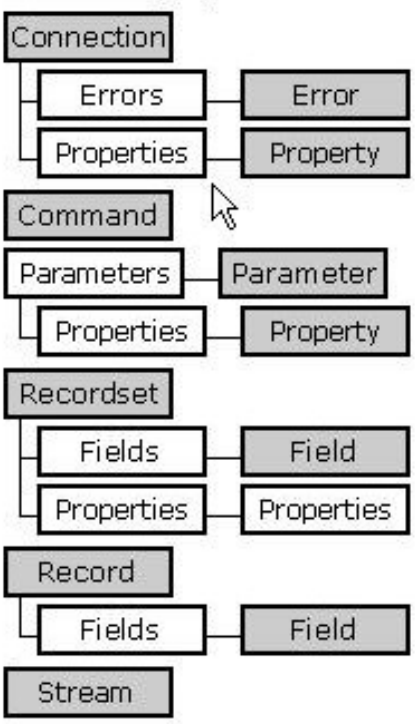


Abbildung 3.2: Das ADO-Objektmodell [Mic06]

Die drei wichtigsten Objekte sollen nachfolgend kurz erläutert werden :

Connection: Stellt die Verbindung zur Datenquelle her. Es muss angegeben werden, um welche Art von Datenquelle es sich handelt, wie die Datenquelle heißt und wo sie liegt.

Command: Wird verwendet, um eine spezifische Abfrage, z.B. ein SQL-Statement, an die verbundene Datenquelle abzusetzen.

Recordset: Eine flexible Struktur, die die eigentlichen Daten enthält und auf der die Bearbeitung der Daten stattfindet (update, insert, delete). Das Recordset dient der Anzeige und Navigation durch Datensätze und gibt Daten in Form von Tabellen und Abfragen zurück.

3.2.2 Oracle Warehouse Builder 10.2.0.1

Der Oracle Warehouse Builder (OWB) ist im weitesten Sinn als Tool zum Daten-Management zu sehen. Es ermöglicht die Bereinigung von Datenbeständen zur Sicherstellung der Datenqualität, Datenüberprüfung, relationales und dimensionales Modellieren der Daten sowie das Management des kompletten Lebenszyklus von Daten und Metadaten. Im Folgenden werden einzelne Funktionalitäten des Warehouse Builders, die für die Bearbeitung dieses Projekts von Bedeutung gewesen wären, kurz erläutert. Eine vollständige Beschreibung der umfangreichen Funktionalitäten dieser Software hätte den Rahmen dieser Arbeit gesprengt.

Entwickelt wurde die Software zum Aufbau eines Data-Warehouses. Darunter kann ganz allgemein eine zentrale Datensammlung verstanden werden. Hauptzweck eines Data-Warehouses ist die Zusammenführung von Daten aus den unterschiedlichen Bereichen eines Unternehmens, um übergreifende Auswertungen zu ermöglichen.
Die Integration von Daten in ein Data-Warehouse-System erfolgt im Rahmen von sogenannten ETL-Prozessen, die von der Software bereitgestellt werden. ETL steht für **E**xtract-**T**ransform-**L**oad. Der Prozess der Datenintegration lässt sich also in die folgenden drei Phasen unterteilen:

1. Extraktion ausgewählter Daten aus verschiedenen Datenquellen
2. Transformation der Daten, die aus unterschiedlich strukturierten Quellen stammen können
3. Laden der bereinigten Daten in das Data-Warehouse

Der komplexeste Teil des ETL-Prozesses ist die Transformation. Die Daten müssen vor dem Laden in das Zielsystem an dessen Strukturen angepasst und gleichzeitig bereinigt werden. Zur Datenbereinigung zählt z.B. die Anpassung von Datentypen, die Auflösung von Verschlüsselungen und die Eliminierung von Duplikaten. Die Verhinderung der redundaten Speicherung von Daten spielt bei der Zusammenführung aus verschiedenen Quellsystemen eine wichtige Rolle. Eine automatisierte Löschung der Duplikate sollte jedoch kritisch betrachtet werden. Grundsätzlich sollten Datenlöschungen immer fachlich beurteilt werden. Gibt es z.B. in verschiedenen Datenquellen eine Person mit dem Namen Hans Maier, die aber jeweils eine andere ID besitzt, so kann das System nicht entscheiden, ob es sich in beiden Quellen um die gleiche Person oder um verschiedene Personen mit dem Namen Hans Maier handelt.

Um den Oracle Warehouse Builder einsetzen zu können, muss eine Oracle Datenbank vorhanden sein. Im ITZ wird die relationale Datenbank „Oracle Database 10g, Release 2 (10.2), Enterprise Edition“ verwendet, die Mitte 2005 von der Firma Oracle auf den Markt gebracht wurde. Die Software zum Aufsetzen der Datenbank war im ITZ bereits vorhanden. Um den Warehouse Builder testen zu können, ist es sinnvoll, eine lokale Oracle Datenbank aufzusetzen, da diese im Fall eines Crashs schnell wieder gelöscht und neu aufgesetzt werden kann. Außerdem wird damit sichergestellt, dass durch den Test keine anderen wichtigen Datenbanken beschädigt werden.
Die Hauptapplikation des Warehouse Builders ist das in Abbildung 3.3 dargestellte „Design Center“, das eine grafische Benutzeroberfläche bereitstellt, auf der Datensysteme entwickelt und visualisiert werden können. Von hier aus sind alle Komponenten des Warehouse Builders zugänglich.

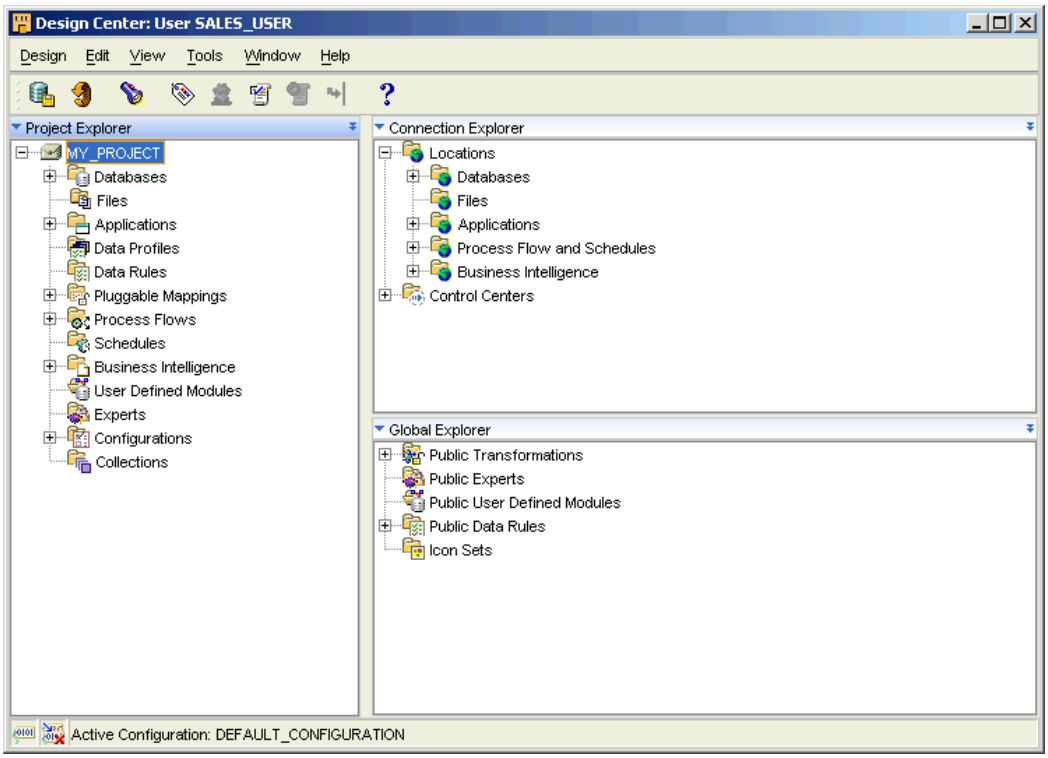


Abbildung 3.3: Das Design Center des Oracle Warehouse Builders [Ora06]

Da der Wert der Daten durch eine hohe Datenqualität steigt, ist es besonders wichtig, diese zu überprüfen. Eine automatisierte Datenqualitätskontrolle soll laut Oracle über die Data-Profiling-Komponente des Warehouse Builders möglich sein. Allerdings ist zu beachten, dass nicht alle Daten „Data Profiling-tauglich“ sind. Zudem muss die Hardware Umgebung für eine Data-Profiling-Analyse stimmen (z.B. Arbeitsspeichergröße etc.) [Ora05].

3.3 Bewertung der Werkzeuge

In diesem Kapitel soll dargestellt werden, wo jeweils die Vor- und Nachteile der beiden Möglichkeiten zur Entwicklung des Konvertierungswerkzeugs liegen, und begründet werden, warum der Programmierung mit Visual Basic der Vorzug gegeben wurde.
Ausschlaggebend für die Entscheidung, das Werkzeug selbst zu programmieren, war die Tatsache, dass für die Bearbeitung des Themas nur ein begrenzter Zeitraum zur Verfügung steht. Die tiefere Einarbeitung in die Software Warehouse Builder und deren Funktionalitäten wurde nach der Auseinandersetzung mit verschiedenen Handbüchern und technischen Berichten als zu umfangreich für dieses Projekt erachtet. Bereits die Einrichtung der Datenbank Oracle 10g, die zum Testen lokal installiert werden muss sowie die Installation und Konfiguration des Warehouse Builders, erfordern einen Zeitaufwand, der für dieses Projekt als zu hoch eingestuft wurde. Vorteilhaft ist am Warehouse Builder sicherlich, dass ein Projekt, das den Export/Import von Daten zum Thema hat, grafisch in visuellen Editoren bearbeitet werden kann. Die Software stellt zudem eine Bibliothek mit vordefinierten Transformationen zur Verfügung.

Die eigene Programmierung des Werkzeugs bietet zahlreiche Vorteile. Mit entsprechenden Programmierkenntnissen, die im ITZ vorhanden sind, kann eine solche Anwendung recht einfach und schnell entwickelt werden. Zudem ist die Programmierung flexibler, da Sonderfälle abfangbar sind und der Quellcode an die Anforderungen der jeweiligen Datenquelle angepasst werden kann. Das Programm kann von jedem Anwender an jedem beliebigen Arbeitsplatz ausgeführt werden, da keine Komponenten installiert werden müssen. Es ist zudem durch die einfach gestaltete, grafische Anwendungsoberfläche leicht zu handhaben und ohne Einarbeitung verwendbar.
Betrachtet man nun den Wert einer solchen Programmierung für den Import weiterer Datenquellen, so lässt sich sagen, dass der Quellcode durchaus als Grundgerüst für eine neue Programmierung verwendet werden kann. Natürlich muss dieser dann an die Struktur der jeweiligen Datenquelle angepasst werden. Dies sollte aber ohne zu große Probleme realisierbar sein, da in den VB-Code viele SQL-Statements, die meist auf Views zugreifen, eingebettet wurden. Sowohl vor Verwendung des Warehouse Builders, wie auch vor Beginn einer Programmierung ist die Analyse der Datenquellenstruktur jedoch unerlässlich.
Als Nachteil der Programmierlösung gegenüber der Softwarelösung kann die Tatsache erachtet werden, dass die Datenqualität hier nicht automatisiert beim Import überprüft werden kann. Allerdings müssen auch dem Warehouse Builder Regeln für die Datenqualitätskontrolle vorgegeben werden, die nur von der Fachabteilung bestimmt werden können.

Nach sorgfältiger Abwägung der dargestellten Vor- und Nachteile der beiden Lösungen, wurde die Entscheidung zugunsten der Programmierung mit Visual Basic getroffen. Im folgenden Kapitel soll diese Lösung detailliert dargestellt werden.

4 Programmierung des Konverters

In den folgenden Unterkapiteln soll die Vorgehensweise bei der Programmierung des Konvertierungswerkzeugs mit Visual Basic beschrieben werden. Es wurde versucht, die Beschreibung der Vorgehensweise möglichst einfach zu halten und sie durch Einbindung von Quellcode-Ausschnitten zu verdeutlichen. Der vollständige Programmcode ist im Anhang finden. Zur Veranschaulichung befindet sich dort auch ein Programmablaufdiagramm, das aus Platzgründen in die verschiedenen Funktionen unterteilt wurde.

Zur Programmierung des Konverters in Visual Basic wird in der Entwicklungsumgebung Microsoft Visual Studio 6.0 ein Modul in einem neuen Projekt angelegt, in das dann der Programmcode geschrieben wird. Die Daten der zuvor ausgewählten Tabellen aus dem Schema ALBIS müssen in folgender Reihenfolge in die entsprechenden Tabellen des Schemas ARTIS importiert werden:

1. Daten aus der Tabelle B24A_PERSON in die Tabelle AE_PERSON
2. Daten aus der Tabelle B24A_BIOTOP in die Tabelle AE_FUNDORT
3. Daten aus der Tabelle B24A_ BIOTOP in die Tabelle AE_ARTENFUND
4. Daten aus der Tabelle B24A_AB_BIOTOP in die Tabelle AE_ARTENFUND und die Tabelle AE_ZUORD_AF_HAEU

Es muss zwingend mit dem Import der Personendaten begonnen werden, da die eindeutige Personennummer aus der Tabelle AE_PERSON in den Tabellen AE_FUNDORT und AE_ARTENFUND als Fremdschlüssel verwendet wird und diese daher erst vorhanden sein muss, bevor weitere Daten in die anderen Tabellen eingefügt werden können.

4.1 Der Aufbau des Konvertierungsprogramms

Um den Ablauf eines Importvorgangs darzustellen, sollen hier zunächst alle Funktionen entsprechend der Reihenfolge ihres Aufrufs im Programm erläutert werden. Um eine verständliche Erläuterung des Programmcodes zu liefern, lässt es sich an einigen Stellen jedoch nicht vermeiden, auf noch nicht dargestellte Funktionen vorzugreifen oder bereits Erwähntes zu wiederholen. Zur Benennung der Funktionen lässt sich allgemein sagen:

- Funktionen, die Werte aus der Quelltabelle abrufen, sind am *get* im Funktionsnamen zu erkennen.
- Funktionen, welche Werte in eine Zieltabelle schreiben, sind am *getNew* im Funktionsnamen zu erkennen.

4.1.1 Allgemeiner Deklarationsteil und Hauptprozedur

Zu Beginn des Skripts werden im allgemeinen Deklarationsteil zunächst alle benötigten globalen Variablen definiert. Unter einer globalen Variablen versteht man eine Variable, die ab dem Zeitpunkt ihrer Deklaration für das ganze Programm Gültigkeit hat. Im Gegensatz dazu gibt es auch lokale Variablen, die nur in der Funktion, in der sie deklariert wurden, verwendet werden können.

```
Public dbQuelle As New ADODB.Connection
Public dbZiel As New ADODB.Connection
Public countNewPerson, countDatensaetze, countNewArtenfund,
   countNewFundort, countNewArtstatus_ID, anzahlDatensaetze As Double
Public bolErrorOccured As Boolean
Public Abbrechen As Boolean
```

Wie aus folgendem Quellcode ersichtlich, wird in der Hauptprozedur der generelle Programmablauf beschrieben:

```
Public Sub Main()
  On Error GoTo ErrorHandler

   Call openDB
   Load Import_starten
   Import_starten.Show

 Exit Sub
```

Dort wird zunächst die Funktion *openDB()* aufgerufen, welche die Datenbankverbindungen zu den Schemata ALBIS und ARTIS herstellt. War dies erfolgreich, wird die Anwendungsoberfläche über `Load Import_starten` geladen und über das Schlüsselwort „Show“ dem Benutzer angezeigt.

4.1.2 Die Funktion OpenDB()

Die Erstellung der Verbindungen zu den Datenbankschemata erfordert die Verwendung der in Kapitel 3.2.1 beschriebenen Objekte des ADO-Objektmodells. Die Verbindung zur Datenbank erfolgt über den OLE DB-Provider für Oracle und eine ODBC-Schnittstelle. Die Verbindungsparameter (Properties), die notwendig sind, um eine Verbindung zu einer Oracle Datenbank aufzubauen, können dem nachfolgenden Quellcode-Ausschnitt entnommen werden.

```
Public Sub openDB()
  On Error GoTo ErrorHandler

dbQuelle.Provider = "OraOLEDB.Oracle.1"
dbQuelle.Properties("Data Source") = "UIST2_DB"
dbQuelle.Properties("Persist Security Info") = "False"
dbQuelle.Properties("User ID") = "albis"
dbQuelle.Properties("Password") = "***"
dbQuelle.Open
dbQuelle.CursorLocation = adUseClient

dbZiel.Provider = "OraOLEDB.Oracle.1"
dbZiel.Properties("Data Source") = "UIST2_DB"
```

```
dbZiel.Properties("Persist Security Info") = "False"
dbZiel.Properties("User ID") = "artis"
dbZiel.Properties("Password") = "xxx"
dbZiel.Open
dbZiel.CursorLocation = adUseClient
```

4.1.3 Der Start der Anwendungsoberfläche

Nach dem Laden der Anwendungsoberfläche wird dem Formularmodul mit dem Namen „Import_starten“ die Kontrolle über den weiteren Programmablauf übergeben. Der Quellcode dieses Moduls befindet sich auf den letzten beiden Seiten von Anhang B. In diesem Modul werden zunächst die vom Benutzer getätigten Eingaben in die beiden Textfelder geprüft. Werden diese als korrekt eingestuft, wird die *importDB()*-Funktion des Standardmoduls aufgerufen. Abbildung 4.1 zeigt die Anwendungsoberfläche mit den beiden Textfeldern für die Eingabe der Biotopnummern, den Schaltflächen zur Prüfung der beiden Nummern, den Start des Importvorgangs, das Beenden der Anwendung, der Schaltfläche zum Löschen der Eingaben und zum Abbrechen des Importvorgangs.

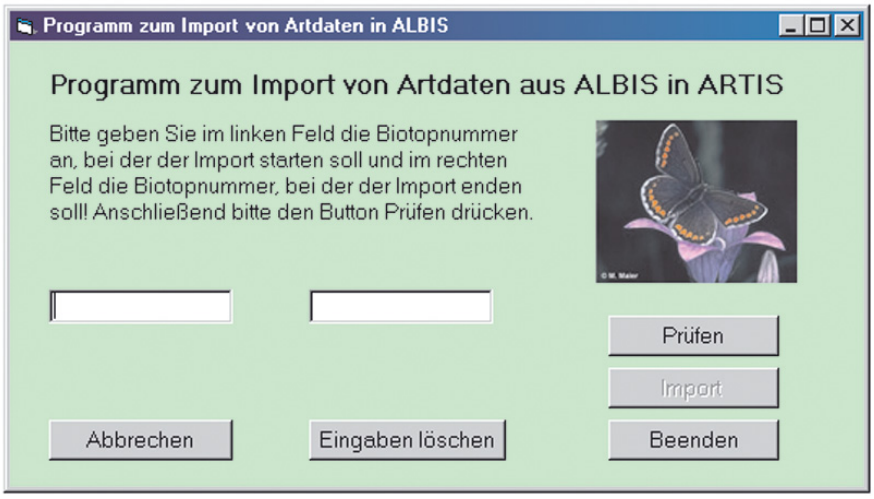


Abbildung 4.1: Anwendungsoberfläche des Importwerkzeugs

Da die Prüfung der Benutzereingaben eine wesentliche Rolle für den korrekten Ablauf des Importvorgangs spielt, soll sie an dieser Stelle kurz erläutert werden. Es wird zunächst geprüft, ob in beide Felder überhaupt etwas eingetragen wurde und im nächsten Schritt, ob es sich auch um numerische Werte handelt. Dann erst wird über einen Abgleich mit der Datenbank kontrolliert, ob es sich bei den eingegebenen Werten um gültige Biotopnummern handelt. Ist mindestens eine der beiden Nummern nicht gültig, wird dem Benutzer die Meldung ausgegeben, dass die Nummer nicht gültig ist und er diese korrigieren soll. Die Prüfung wird solange fortgesetzt, bis beide Nummern als korrekt eingestuft werden.
Außerdem wurde eine Funktion integriert, die feststellt, ob in das linke Feld auch die kleinere der beiden Nummern eingegeben wurde. Ist dies nicht der Fall, werden die Nummern automatisch vertauscht. Die Eingabe der kleineren Nummer in das linke Feld ist erforderlich, da

der Import der Artdaten aufsteigend nach Biotopnummern – in Datensatzblöcke unterteilt – erfolgen soll.
Erst wenn alle Angaben richtig sind, wird die Schaltfläche `Import` freigegeben und der Importvorgang kann vom Anwender gestartet werden. Anhand eines „Fortschrittbalkens“, der unterhalb der beiden Eingabefelder angebracht wurde, kann während des Importvorgangs nachvollzogen werden, wie weit der Import bereits fortgeschritten ist. Dort erscheint dann eine Meldung in der Art: „`Datensatz 10 von 540 wird gerade importiert.`“

4.1.4 Die Funktion importDB()

In dieser Funktion müssen zunächst alle Variablen deklariert werden, die den weiteren Funktionen beim Aufruf mitgegeben werden sollen. Außerdem wird ein Recordset *rsQuelle* definiert, in dem die Ergebnisse der *select*-Abfrage gespeichert werden.
Zu Beginn dieser Funktion wird zunächst die letzte vorhandene Import_ID aus der Tabelle AE_CTR_IMPORT abgerufen und dann die nächstgrößere Import_ID erzeugt. Diese wird immer nur zu Beginn eines Importvorgangs erstellt und bleibt innerhalb des ganzen Programmdurchlaufs gleich. Die neue Import_ID wird mit folgendem Code erzeugt:

```
rsID.Open "SELECT (MAX(IMPORT_ID)+1)AS newImportID FROM AE_CTR_IMPORT",
           dbZiel
   getNewImport_ID = rsID("newImportID")
rsID.Close
```

Diese ID wird benötigt, um in der Importtabelle des ARTIS-Schemas (AE_CTR_IMPORT) den Importvorgang zuordnen zu können. Der Zweck der Importtabelle wird in Kapitel 4.2 genauer erläutert.
Die eigentliche Datenabfrage aus dem Schema ALBIS beginnt dann mit einer select-Abfrage auf die View B24AVN_BIOTOP_AB_BIOTOP:

```
rsQuelle.Open "SELECT * FROM B24AVN_BIOTOP_AB_BIOTOP WHERE AB_BNUMG
 between '" + von + "' and  '" + bis + "' order by ab_bnumg,
 qpmukey desc nulls last", dbQuelle
```

Die vom Benutzer in die Eingabefelder eingetragenen Werte werden an Stelle der Variablen „von“ bzw. „bis“ in die Abfrage eingesetzt.

4.1.5 Die Funktionen getPerson_Nr() und getNewPersonNr()

Der Funktion *getPerson_Nr()* werden beim Aufruf die Biotopnummer (*ab_bnumg*), die zugehörigen Artnummern und die Beobachtungsdaten übergeben. Um eine Fehlermeldung zu vermeiden, muss zunächst geprüft werden, ob der Kombination aus Biotopnummer, Artnummer und Beobachtungsdatum überhaupt ein Eintrag in der Personentabelle zugeordnet ist. Trifft dies nicht zu, so wird die Zahl 0 zurückgegeben.
Liefert die Abfrage ein Ergebnis zurück, wird weiter überprüft, ob in dieser Zeile eine Personennummer (p_mukey) steht. Ist dies nicht der Fall, wird ebenfalls eine 0 zurückgegeben. Stehen in beiden Feldern Werte, wird anhand der eindeutigen Kombination aus Nachname, Vorname und Kennung überprüft, ob die Person schon in der Zieltabelle AE_PERSON existiert. Diese Überprüfung ist notwendig, um redundante Einträge in der Personentabelle zu vermeiden.

Ist die Person noch nicht vorhanden, werden der Funktion *getNewPerson_Nr()* die zur Eintragung in die Zieltabelle bestimmten Werte beim Aufruf übergeben. Gibt es die Person in der Zieltabelle bereits, wird die vorhandene Personennummer abgerufen und in den weiteren Funktionen verwendet. Außerdem wird in diesem Fall in die Log-Datei die Meldung „`Person mit der PERSON_NR x existiert schon. Vorhandene PERSON_NR wird verwendet!`“ geschrieben.

In der Funktion *getNewPerson_Nr()* wird die Abfrage der höchsten vorhandenen Personennummer und die Erzeugung der nächstgrößeren realisiert. Diese wird dann als neue ID für den Eintrag des Datensatzes verwendet. Der Quellcode lautet wie folgt:

```
rsID.Open "SELECT (MAX(PERSON_NR)+1) as newID FROM AE_PERSON", dbZiel
     getNewPerson_Nr = rsID("newID")
rsID.Close
```

Dieser Codeteil wird hier und in allen weiteren Funktionen, die Datensätze in die Zieltabellen schreiben, in ähnlicher Weise verwendet, da beim Eintrag eines neuen Datensatzes jeweils eine neue laufende Nummer erzeugt und verwendet werden soll. Damit ist sichergestellt, dass der Datensatz eindeutig identifizierbar ist. Nach dem Öffnen der Zieltabelle werden die abgefragten Werte in die Tabelle AE_PERSON geschrieben und die Tabelle danach über *Update* aktualisiert. Als Beispiel für die Eintragung von Daten in eine Zieltabelle kann der nachfolgend aufgeführte Quellcode gelten:

```
rsZiel.AddNew

rsZiel("PERSON_NR") = getNewPerson_Nr
rsZiel("NACHNAME") = nachname
rsZiel("VORNAME") = vorname
rsZiel("ANREDE") = anrede
rsZiel("KENNUNG") = 0
rsZiel("TITEL") = titel
rsZiel("STRASSE") = strasse
rsZiel("POSTFACH") = postfach
rsZiel("TELEFON") = telefon
rsZiel("FAX") = fax
rsZiel("ORT") = wohnort

rsZiel.Update
rsZiel.Close

Call write_ctr_Import("1", CStr(p_ukey), getNewPerson_Nr)
```

In der letzten Zeile wird hier über das Schlüsselwort *Call* die Funktion *write_ctr_Import()* aufgerufen. Diese trägt die Werte für die *tab_id*, die *alt_nr* und die *neu_nr* in die CTR_IMPORT Tabelle von ARTIS ein. Ausführliche Erläuterungen zu dieser Funktion sind in Kapitel 4.2 zu finden.

Eine wichtige Rolle bei der Erzeugung einer syntaktisch korrekten SQL-Abfrage spielt die Funktion *convertNullToSqlNull()*. Sie wird innerhalb der Funktion *getPerson_Nr()* an folgender Stelle verwendet:

```
rsZiel.Open "SELECT * FROM AE_PERSON WHERE NACHNAME " &
convertNullToSqlNull(rsQuelle("P_NNAME")) & " AND VORNAME " &
convertNullToSqlNull(rsQuelle("P_VNAME")) & " AND KENNUNG=0", dbZiel
```

Ist in den Spalten *p_nname* oder *p_vname* im betreffenden Feld kein Eintrag vorhanden – der Wert wäre damit NULL – würde ohne die Zusatzfunktion *convertNullToSqlNull()* keine syntaktisch korrekte SQL-Abfrage zustande kommen und eine Fehlermeldung erzeugt werden. Die Syntax der *convertNullToSqlNull()* Funktion lautet wie folgt:

```
Private Function convertNullToSqlNull(x) As String
    If IsNull(x) Then convertNullToSqlNull = "IS NULL"
                 Else
    convertNullToSqlNull = "='" + CStr(x) + "'"
End Function
```

4.1.6 Die Funktionen getFundort_Nr() und getNewFundort_Nr()

Die Funktionen *getFundort_Nr()* und *getNewFundort_Nr()* dienen der Abfrage der Fundortdaten aus der Quelltabelle bzw. der Eintragung jener in die Zieltabelle.
Damit der Fundort innerhalb eines Importvorgangs nicht für jede im Biotop vorkommende Art nochmals eingetragen wird, wurde eine bedingte Abfrage vor jeden Aufruf der Funktion *getFundort_Nr()* geschaltet. Diese prüft, ob sich die Biotopnummer und die Personennummer gegenüber dem vorherigen Durchlauf verändert haben. Diese Überprüfung ist nur möglich, da die in das Recordset rsQuelle gespeicherten Ergebnisse der Abfrage auf die View B24AVN_BIOTOP_AB_BIOTOP aufsteigend nach Biotopnummer (*ab_bnumg*) und nachfolgend nach der Personennummer (*qpmukey*) geordnet wurden. Hat sich entweder die Biotopnummer oder bei gleichbleibender Biotopnummer die Personennummer verändert, wird eine neue Fundortnummer vergeben.
Bevor die Fundortdaten jedoch in die Tabelle AE_FUNDORT eingetragen werden können, muss überprüft werden, ob der Fundort bereits in der Zieltabelle existiert. Die Prüfung muss in diesem Fall jedoch anders ablaufen als die Überprüfung der Zieltabelleneinträge in der Funktion *getPersonNr()*. Dies liegt im Aufbau der Zieltabelle begründet. In der Tabelle AE_FUNDORT ist nämlich keine Spalte zur Speicherung der Biotopnummer vorgesehen. Die Verbindung zwischen der Biotopnummer und der neu vergebenen Fundortnummer wird nur durch die CTR_IMPORT Tabelle geschaffen. In diesem Fall wird in die Spalte *alt_nr* die Biotopnummer (*ab_bnumg* aus der Quelltabelle) geschrieben. Somit muss die Prüfung, ob der Fundort bereits in ARTIS angelegt wurde, auf das entsprechende Feld der Spalte *alt_nr* dieser Tabelle und nicht auf ein Feld der Tabelle AE_FUNDORT stattfinden. Existiert der Fundort bereits, wird die vorhandene Fundortnummer aus der Spalte *neu_nr* der CTR_IMPORT-Tabelle abgerufen und als *fundort_nr* in den weiteren Funktionen verwendet. Muss kein neuer Fundort angelegt werden, wird die Funktion *getNewFundort_Nr()* auch nicht mehr aufgerufen.
Die in die Tabelle AE_FUNDORT zu importierenden Daten stammen aus mehreren Quelltabellen. Um die im Programmcode verwendeten SQL-Abfragen möglichst einfach halten zu können, wurden diese Daten in der View B24AVN_BIOTOP_AB_BIOTOP auf dem Schema ALBIS zusammengefasst. Diese ist wie folgt aufgebaut:

```
CREATE OR REPLACE VIEW b24avn_biotop_ab_biotop
(ab_bnumg,ab_alfunrn,art_beob_dat, bio_name,tk_nr,bio_r, bio_h,
 bemerkung, bio_aenddat, bio_erfdat,geologie_id, geo_name,
 status_id, qpmukey, tk_quadrant)
AS SELECT DISTINCT ab.ab_bnumg, ab.ab_alfunrn, ab.ab_jahr, b.bio_name,
b.bio_numtk, b.bio_r,b.bio_h,ab.ab_bem, b.bio_beadat,b.bio_erfdat,
c.igb_geolfu, d.geo_name, ab.ab_status, ab.ab_qpmukey,
DECODE (qdr,'NW', '1',
            'NO', '2',
            'SW', '3',
            'SO', '4') tk_quadrant

FROM b24a_ab_biotop ab,
     b24a_biotop b,
     b24a_igeobio c,
     b24a_geologie d,
     b24avn_zuord_bio_tk25 e,
     b24avn_person f

WHERE
ab.ab_bnumg = b.bio_numges
AND ab.ab_bnumg = e.bio_numges
AND c.igb_bnumg(+) = b.bio_numges
AND c.igb_geolfu = d.geo_lfu(+)
AND ab.ab_qpmukey = f.p_mukey(+)
```

Ist der Fundort in der Zieltabelle noch nicht vorhanden, wird die Funktion *getNewFundort_Nr()* aufgerufen und die abgefragten Werte übergeben. Der Aufruf dieser Funktion lautet wie folgt:

```
getFundort_Nr = getNewFundort_Nr(rsQuelle("BIO_NAME"),
                rsQuelle("TK_Nr"), rsQuelle("TK_QUADRANT"),
                rsQuelle("BIO_R"), rsQuelle("BIO_H"),
                rsQuelle("GEOLOGIE_ID"), rsQuelle("BIO_ERFDAT"),
                rsQuelle("BIO_AENDDAT"), l_biotop_nr, l_Person_Nr)
```

Die Werte der jeweils in Klammer hinter rsQuelle angegebenen Spaltenbezeichnungen werden über die Funktion *getNewFundort_Nr()* in die Tabelle AE_FUNDORT gespeichert. Um zu verhindern, dass beim Öffnen der Zieldatei alle Datensätze in den Speicher geladen werden, wird in der Funktion *getNewFundort_Nr()* zusätzlich die Abfrage:

```
rsZiel.Open "SELECT * FROM AE_FUNDORT WHERE FUNDORT_NR=0"
```

eingefügt. Ähnliche Abfragen werden auch in den anderen Funktionen, in denen Daten in die Zieltabellen eingetragen werden, verwendet. Diese Maßnahme dient der Verbesserung der Ablaufgeschwindigkeit des Programms.

4.1.7 Die Funktionen getArtenfund_Nr() und getNewArtenfund_Nr()

Informationen zu den Artenfunden werden über die Funktion *getArtenfund_Nr()* aus dem Quellschema abgerufen. Diese Funktion ist ähnlich wie die beiden anderen zuvor aufgebaut und soll daher nicht mehr bis ins Detail beschrieben werden.

In dieser Funktion muss vor dem Import überprüft werden, ob der Artenfund an diesem Fundort schon existiert. Im Gegensatz zu den bereits besprochenen Funktionen ist es bei dieser Funktion jedoch nicht möglich, die Überprüfung der Einträge der Zieltabelle nur anhand eines Feldes zu machen. Prinzipiell sind die Arten mehrfach in der Tabelle AE_ARTENFUND vorhanden. Sie sollten grundsätzlich auch für jeden Fundort nur einmal gespeichert werden. Nun tritt aber der Fall auf, dass die gleiche Art am gleichen Fundort mehrfach vorkommt, jedoch mit einem unterschiedlichen Beobachtungsdatum. Daher muss die Überprüfung um das Beobachtungsdatum erweitert werden:

```
"SELECT * FROM AE_ARTENFUND WHERE FUNDORT_NR ='" + l_fundort_nr + "'
and LFUNRN = '" + l_LFUNRN + "' and
BEOBACHTUNGSDATUM = '" + art_beob_dat + "'"
```

Ist das Ergebnis der Abfrage NULL, bedeutet dies, dass der Artenfund an diesem Fundort mit dem gewählten Beobachtungsdatum noch nicht vorhanden ist und eingetragen werden muss. Bekommt die Abfrage ein Ergebnis zurückgeliefert, wird in die Log-Datei geschrieben, dass der Artenfund mit dieser Nummer an diesem Fundort mit diesem Beobachtungsdatum schon existiert. Außerdem wird die vorhandene Artenfundnummer abgerufen und in den weiteren Funktionen, z.B. zur Übergabe an die Funktion *getArtstatus_ID()*, verwendet.
Der Funktion *getNewArtenfund_Nr()* muss neben der Personennummer (l_person_nr), der Fundortnummer (l_fundort_nr) und der Artnummer (l_lfunrn) auch noch das Beobachtungsdatum (art_beob_dat) übergegeben werden.

Wie sich erst bei den ersten Importversuchen herausstellte, können am gleichen Fundort auch Artenfunde mit gleicher Artenfundnummer und gleichem Beobachtungsdatum, aber unterschiedlicher Geologie_ID existieren. Dies kommt dadurch zustande, dass ein Fundort auf (hier maximal zwei) unterschiedlichen geologischen Einheiten liegen kann und der Kartierer beide Informationen eintragen wollte. Bedingt durch die Tatsache, dass die Fundortnummer der Primärschlüssel ist, kann für diesen Fundort leider nur ein Datensatz mit einer Geologie_ID übernommen werden. Da jedoch nicht festgelegt werden kann, welche die größere oder bedeutendere geologische Einheit ist, muss in diesem Fall dem Programm überlassen werden, welcher Datensatz mit welcher Geologie_ID übernommen wird. Diese Problematik sollte möglichst mit der Fachabteilung geklärt und eine Entscheidung getroffen werden, wie zukünftig mit dieser Tatsache umzugehen ist.

4.1.8 Die Funktionen getArtstatus_ID() und getNewArtstatus_ID()

Vom Grundgerüst ebenfalls ähnlich aufgebaut wie die vorherige Funktion ist die Funktion *getArtstatus_ID()*. Die Zieltabelle für die Speicherung der Informationen über den Status einer Art ist die Tabelle AE_ZUORD_AF_HAEU. Unter dem Status einer Tier- oder Pflanzenart werden zusätzliche Informationen zur „Lebenssituation“ verstanden wie z.B. die Angaben, ob die Art unter Brutverdacht steht, ob es sich um einen Durchzügler handelt oder ob die Pflanze austreibend ist.
Die Tabelle ARTIS.AE_ZUORD_AF_HAEU hat ihre Struktur dem Artenerfassungsprogramm zu verdanken. Der Primärschlüssel ist hier aus vier Feldern (*artenfund_nr*, *artstatus_id*, *haeufigkeitstyp_id* und *haeufigkeit_id*) zusammengesetzt. In die Felder, die den Primärschlüssel bilden, müssen generell immer Daten eingetragen werden, da über sie die Eindeutigkeit der Datensätze sichergestellt wird.

Ebenfalls nicht NULL sein dürfen die Felder der Spalten *herba*, *kartei* und *literatur*. Eine 0 in diesen Spalten bedeutet, dass es sich nicht um eine Information aus einem Herbarium, einer Kartei oder aus einer Literaturquelle handelt, eine 1 bedeutet entsprechend das Gegenteil. Da die § 24a-Biotopkartierung als Felderhebung durchgeführt wurde und die Daten neu erfasst wurden, muss an diesen Stellen wie aus dem folgenden Quellcodeausschnitt ersichtlich, immer eine 0 eingefügt werden.

```
rsZiel("ARTENFUND_NR") = l_Artenfund_Nr
  If IsNull(AB_STATUS) Then
        rsZiel("ARTSTATUS_ID") = 0
  Else
        rsZiel("ARTSTATUS_ID") = AB_STATUS
rsZiel("HAEUFIGKEITSTYP_ID") = "ga"
rsZiel("HAEUFIGKEIT_ID") = AB_MENGE
rsZiel("HERBA") = 0
rsZiel("KARTEI") = 0
rsZiel("LITERATUR") = 0
```

In der Spalte *haeufigkeitstyp_id* wird definiert, um was für einen Häufigkeitstyp es sich handelt. Zur Entschlüsselung der Werte wurde auf dem Schema ARTIS die Schlüsseltabelle AE_SL_HAEUFIGKEITSTYP geschaffen. Für die § 24a-Daten wird hier immer das Kürzel „ga“ eingetragen. Dieses steht für „Menge (aus § 24a-Kartierung und Artenkataster Baden-Württemberg)“.
In die Spalte *haeufigkeit_id* werden die Angaben aus der ALBIS Tabellenspalte *ab_menge* übernommen.
Da in der Spalte *artstatus_id* der Tabelle AE_ZUORD_AF_HAEU keine NULL-Werte akzeptiert werden, wird zunächst durch eine Abfrage auf die Quelltabelle überprüft, ob im Feld *ab_status* ein Eintrag vorhanden ist. Ist kein Eintrag vorhanden (der Wert ist also NULL), wird die Zieltabelle geöffnet und überprüft, ob die Artenfundnummer bereits in der Tabelle existiert. Wird zu dieser Artenfundnummer kein Eintrag in der Zieltabelle gefunden, wird dem Aufruf der Funktion *getNewArtstatus_ID()* die Funktion *convertNullToNumber()* vorgeschaltet. Diese dient dazu, den Wert NULL in die Zahl 0 umzuwandeln. Die umgewandelten Werte werden dann an die Funktion *getNewArtstatus_ID()* übergeben und können durch sie problemlos in die Zieltabelle eingetragen werden. Eine 0 hat in einer Datenbank meist die Bedeutung „keine Angabe vorhanden“. Somit ist diese Umwandlung auch fachlich korrekt. Sie ist nötig, da sonst eine Constraint-Verletzung auftritt, die den Import des Datensatzes in die Zieltabelle verhindert. Mit einem Constraint soll hier sichergestellt werden, dass kein Wert in ein Feld dieser Spalte eingetragen wird, der nicht über die Schlüsseltabelle zu entschlüsseln ist. Sinn dieses Constraints ist es also die referenzielle Integrität der Datenbank zu wahren. Der Code der Konvertierungsfunktion lautet wie folgt:

```
Private Function convertNullToNumber(X) As String

 If IsNull(X) Then
        convertNullToNumber = "0"
 Else
        convertNullToNumber = CStr(X)

Exit Function
```

Steht in der Quelltabelle im Feld *ab_status* ein Wert, wird die if-Anweisung übersprungen und die else-Anweisung ausgeführt. Auch hier muss zunächst überprüft werden, ob der Datensatz in der Zieltabelle bereits existiert. Tut er das nicht, soll er eingetragen werden. Der aus der Quelltabelle abgefragte Wert muss vor der Übergabe an die Funktion *getNewArtstatus_ID()* noch mit Hilfe einer weiteren Funktion bearbeitet werden, da die Schlüsseltabellen von Quell- und Zieltabelle nicht identisch sind. Dies ist notwendig, da *einer* Tabelle nicht verschiedene Schlüsseltabellen zugewiesen werden können und die vorhandene Schlüsseltabelle in diesem Fall aus fachlichen Gründen auch nicht erweiterbar ist. Die Umschlüsselung wird durch nachfolgend dargestellte Funktion *convertNumberToLetter()* realisiert:

```
Private Function convertNumberToLetter(X) As String
  On Error GoTo ErrorHandler

    If X = 0 Then convertNumberToLetter = "0"
    If X = 1 Then convertNumberToLetter = "IN"
    If X = 2 Then convertNumberToLetter = "SY"
    If X = 3 Then convertNumberToLetter = "3"
    If X = 4 Then convertNumberToLetter = "AN"
    If X = 5 Then convertNumberToLetter = "BR"
    If X = 6 Then convertNumberToLetter = "BV"
    If X = 7 Then convertNumberToLetter = "DU"
    If X = 8 Then convertNumberToLetter = "IR"
    If X = 9 Then convertNumberToLetter = "WI"
    If X = Null Then convertNumberToLetter = "0"
```

Diese Funktion prüft zunächst, um welchen Zahlenwert (X) es sich handelt und setzt diesen dann anschließend in die entsprechende Buchstabenfolge bzw. die Zahlen 0 oder 3 um. Wie aus Tabelle 4.1 ersichtlich, ist diese Vorgehensweise möglich, da die Bedeutung bestimmter Zahlen der Quell-Schlüsseltabelle bestimmten Buchstabenfolgen der Ziel-Schlüsseltabelle entspricht.

4.2 Die Kontrolltabelle AE_CTR_IMPORT

Nach dem Eintrag eines jeden Datensatzes in die jeweilige Tabelle (AE_PERSON, AE_FUNDORT, AE_ARTENFUND oder AE_ZUORD_AF_HAEU) wird im Schema ARTIS in der Tabelle AE_CTR_IMPORT ein Eintrag über den erfolgten Import gemacht. Um später nachvollziehen zu können, in welche Tabelle welcher Datensatz eingetragen wurde, werden bestimmte Angaben in dieser Tabelle gespeichert; um welche es sich dabei handelt, kann Abbildung 4.2 entnommen werden.
Jede Tabelle hat eine eindeutige Kennnummer, die *tab_id*, deren Bedeutung wiederum einer Schlüsseltabelle entnommen werden kann. Bei der *alt_nr* handelt es sich um die eindeutige Nummer der Quelltabelle und bei der *neu_nr* um die neue eindeutige Nummer der Zieltabelle, die in den jeweiligen Funktionen erzeugt wurde.
Die Herkunftsnummer gibt Auskunft über die Datenquelle aus der die Daten stammen. Für die Daten der Biotopkartierung wird immer eine 3 eingetragen. Das Importdatum, welches automatisch vom Betriebssystem abgerufen wird, gibt Aufschluss darüber, wann der Vorgang stattgefunden hat. Die Spalte *vorgang* gibt die Art des Datenmanipulationsvorgangs an. Hier wird immer ein „I“ für „Insert“ eingetragen wird, da es sich hier um einen Importvorgang und nicht etwa um eine Aktualisierung(U=Update) oder eine Korrektur (C=Correction) handelt. Wie in Kapitel 4.1.7 beschrieben, kommt es vor, dass gleiche Artenfunde an einem Fundort

NAIS_SL_ARTSTATUS (Schlüsseltabelle zu ARTIS.AE_ZUORD_AF_HAEU)			SCHL_ART_STATUS_24A (Schlüsseltabelle zu ALBIS.B24A_AB_BIOTOP)	
ARTSTATUS_ID	LANGNAME		KENNUNG	NAME
3	**unbeständig**	≙	**3**	**unbeständig**
AB	abblühend			
AD	adult, erwachsen, ausgewachsen			
AL	alt			
AN	**angesalbt**	≙	**4**	**angesalbt**
AS	abgestorben, abgängig			
AT	austreibend			
AU	aufblühend			
BF	Beuteflug			
BR	**Brutpaar**	≙	**5**	**Brutpaar**
BV	**Brutverdacht**	≙	**6**	**Brutverdacht**
CS	cum sporogones			
DU	**Durchzügler**	≙	**7**	**Durchzügler**
EB	epibryisch			
IG	Imago			
II	Individuum, Einzelpflanze			
IM	immaturus			
IN	**indigen**	≙	**1**	**indigen**
IR	**Irrgast**	≙	**8**	**Irrgast**
JU	juvenil, jung, Jungpflanze			
KE	keimend, Keimling			
ST	steril			
SY	**synanthrop**	≙	**2**	**synanthrop**
TT	Totfund, tot			
VB	verblüht, welk			
VE	Vorkommen erloschen			
WA	Wanderung			
WG	Winteraggregation			
WI	**Wintervogel**	≙	**9**	**Wintervogel**
WS	Winterschlaf, Winterruhe			
SQ	Sommerquartier			
WQ	Winterquartier			
0	**keine Angabe**	≙	**0**	**keine Angabe**

Tabelle 4.1: Vergleich der beiden Schlüsseltabellen von ARTIS.AE_ZUORD_AF_HAEU und ALBIS.B4A_AB_BIOTOP

mehrfach vorkommen. Diese besitzen natürlich die gleiche *alt_nr*. Wird versucht, Artenfunde mit der gleichen *alt_nr* einzufügen, wird eine Fehlermeldung ausgegeben, da eine Constraint-Verletzung auftritt. Ein Primärschlüssel, der aus den Spalten *import_id*, *tab_id* und *alt_nr* gebildet wird, reicht daher nicht mehr aus. Aus diesem Grund wurde der Primärschlüssel der Tabelle um die *neu_nr* erweitert. Diese ist nicht gleich, da Artenfunde mit gleicher Nummer, aber unterschiedlichem Beobachtungsdatum ja jeweils eine andere *neu_nr* erhalten. Da die Importtabelle zur Überprüfung der importierten Daten dienen soll, ist es wichtig, dass dort auch wirklich alle importierten Datensätze aufgeführt werden. Daher war die Ergänzung des Primärschlüssels um die *neu_nr* notwendig.

Row#	IMPORT_ID	TAB_ID	ALT_NR	NEU_NR	HERKUNFT_NR	DATUM	VORGANG
917193	54	3	180224360096	102816	3	29.09.2006	I
917194	54	4	100014	815240	3	29.09.2006	I
917195	54	4	100179	815256	3	29.09.2006	I
917196	54	4	100273	815229	3	29.09.2006	I
917197	54	4	100283	815230	3	29.09.2006	I
917198	54	4	100478	815258	3	29.09.2006	I
917199	54	4	100605	815264	3	29.09.2006	I
917200	54	4	100627	815237	3	29.09.2006	I

Abbildung 4.2: Auszug aus der CTR_IMPORT Tabelle des Schemas ARTIS

4.3 Fehlerbehandlung in Visual Basic

Die Fehlerbehandlung kann in VB über den „ESRI ErrorHandler Generator“ vereinfacht werden. Durch die Aktivierung dieses Zusatzmoduls wird in jede Funktion des Projekts automatisch der Zusatz „On Error GoTo ErrorHandler“ geschrieben. Tritt nun ein Fehler auf, wird in das ErrorHandler-Modul gesprungen, welches ebenfalls durch den ESRI ErrorHandler Generator erzeugt wurde. Damit nun die Fehlermeldung nicht in einem Meldungsfenster ausgegeben, sondern in die Log-Datei geschrieben wird, muss die Prozedur des ErrorHandlers um die Zeile

```
ImportLog (sProcedureName & ": " & sErrDescription)
```

erweitert werden.
Durch den Aufruf der Funktion wird, wie unten ersichtlich, in die Log-Datei der Name der Funktion, der Name des Moduls, in dem der Fehler aufgetreten ist und die Beschreibung des Fehlers geschrieben. So ist für den Anwender erkennbar, welcher Fehler in welcher Prozedur aufgetreten ist. Nachfolgend ein Beispiel für eine Fehlermeldung:

```
getNewArtstatus_ID D:\Diplomarbeit\Importsoftware VB\Code\Module1.bas
ORA-00001: Unique Constraint (ARTIS.AE_ZUORD_AF_HAEU_PK) verletzt
```

4.4 Die Log-Datei

Um den Importvorgang für den Nutzer transparent zu gestalten, wurde eine Log-Datei geschrieben. Bei jedem Aufruf der Funktion *ImportLog()* kann in Klammern angegeben werden, was in diese Datei geschrieben werden soll. So können in der Datei z.B. der Ablauf des Importvorgangs und die Anzahl der importierten Datensätze notiert werden. Zusätzlich werden das Datum und die Uhrzeit der jeweiligen Aktion angegeben. Anhand dieses Protokolls kann der Nutzer auch nachvollziehen, ob der Importvorgang erfolgreich war oder, falls Fehler aufgetreten sind, erkennen, um welche es sich handelt. Abbildung 4.3 zeigt einen Ausschnitt aus der Log-Datei.
War der Importvorgang erfolgreich, wird in der Log-Datei ausgegeben, wieviel Personen, Artenfunde, Fundorte und Artstati im Schema ARTIS neu angelegt wurden.

```
29.11.2006 17:09:07: Datensatz Nr.1: Person mit der PERSON_NR 338 existiert schon.
                                     Vorhandene PERSON_NR wird verwendet!
29.11.2006 17:09:07: Datensatz Nr.1: Neuer Fundort mit FUNDORT_NR= 105421
                                     (Biotopnummer: 166251267285) wurde angelegt.
29.11.2006 17:09:07: Datensatz Nr.1: Neuer Artenfund mit ARTENFUND_NR = 854502 und
                                     LFUNRN = 0100119 und Beobachtungsdatum = 2004 wurde angelegt.
29.11.2006 17:09:07: Datensatz Nr.1: Neue Status_ID = 0 von ARTENFUND_NR = 854502 wurde angelegt.
29.11.2006 17:09:08: Datensatz Nr.2: Person mit der PERSON_NR 338 existiert schon.
                                     Vorhandene PERSON_NR wird verwendet!
29.11.2006 17:09:08: Datensatz Nr.2: Neuer Artenfund mit ARTENFUND_NR = 854503 und
                                     LFUNRN = 0101060 und Beobachtungsdatum = 2004 wurde angelegt.
29.11.2006 17:09:08: Datensatz Nr.2: Neue Status_ID = 0 von ARTENFUND_NR = 854503 wurde angelegt.
29.11.2006 17:09:08: Datensatz Nr.3: Person mit der PERSON_NR 338 existiert schon.
                                     Vorhandene PERSON_NR wird verwendet!
29.11.2006 17:09:08: Datensatz Nr.3: Neuer Artenfund mit ARTENFUND_NR = 854504 und
                                     LFUNRN = 0102218 und Beobachtungsdatum = 2004 wurde angelegt.
29.11.2006 17:09:08: Datensatz Nr.3: Neue Status_ID = 0 von ARTENFUND_NR = 854504 wurde angelegt.
29.11.2006 17:09:08: Datensatz Nr.4: Person mit der PERSON_NR 338 existiert schon.
                                     Vorhandene PERSON_NR wird verwendet!
29.11.2006 17:09:08: Datensatz Nr.4: Neuer Artenfund mit ARTENFUND_NR = 854505 und
                                     LFUNRN = 0101541 und Beobachtungsdatum = 2004 wurde angelegt.
29.11.2006 17:09:09: Datensatz Nr.4: Neue Status_ID = 0 von ARTENFUND_NR = 854505 wurde angelegt.
29.11.2006 17:09:09: Datensatz Nr.5: Person mit der PERSON_NR 338 existiert schon.
                                     Vorhandene PERSON_NR wird verwendet!
29.11.2006 17:09:09: Datensatz Nr.5: Neuer Artenfund mit ARTENFUND_NR = 854506 und
                                     LFUNRN = 0100812 und Beobachtungsdatum = 2004 wurde angelegt.
29.11.2006 17:09:09: Datensatz Nr.5: Neue Status_ID = 0 von ARTENFUND_NR = 854506 wurde angelegt.
29.11.2006 17:09:09: Datensatz Nr.6: Person mit der PERSON_NR 338 existiert schon.
                                     Vorhandene PERSON_NR wird verwendet!
29.11.2006 17:09:09: Datensatz Nr.6: Neuer Artenfund mit ARTENFUND_NR = 854507 und
                                     LFUNRN = 0100557 und Beobachtungsdatum = 2004 wurde angelegt.
29.11.2006 17:09:10: Datensatz Nr.6: Neue Status_ID = 0 von ARTENFUND_NR = 854507 wurde angelegt.
29.11.2006 17:09:10: Datensatz Nr.7: Person mit der PERSON_NR 338 existiert schon.
                                     Vorhandene PERSON_NR wird verwendet!
29.11.2006 17:09:10: Datensatz Nr.7: Neuer Artenfund mit ARTENFUND_NR = 854508 und
                                     LFUNRN = 0100002 und Beobachtungsdatum = 2004 wurde angelegt.
29.11.2006 17:09:10: Datensatz Nr.7: Neue Status_ID = 0 von ARTENFUND_NR = 854508 wurde angelegt.
29.11.2006 17:09:10: Datensatz Nr.8: Person mit der PERSON_NR 338 existiert schon.
                                     Vorhandene PERSON_NR wird verwendet!
29.11.2006 17:09:10: Datensatz Nr.8: Neuer Artenfund mit ARTENFUND_NR = 854509 und
                                     LFUNRN = 0101652 und Beobachtungsdatum = 2004 wurde angelegt.
29.11.2006 17:09:10: Datensatz Nr.8: Neue Status_ID = 0 von ARTENFUND_NR = 854509 wurde angelegt.
29.11.2006 17:09:11: Datensatz Nr.9: Person mit der PERSON_NR 338 existiert schon.
                                     Vorhandene PERSON_NR wird verwendet!
29.11.2006 17:09:11: Datensatz Nr.9: Neuer Artenfund mit ARTENFUND_NR = 854510 und
                                     LFUNRN = 0101994 und Beobachtungsdatum = 2004 wurde angelegt.
29.11.2006 17:09:11: Datensatz Nr.9: Neue Status_ID = 0 von ARTENFUND_NR = 854510 wurde angelegt.
29.11.2006 17:09:11: Datensatz Nr.10: Person mit der PERSON_NR 338 existiert schon.
                                      Vorhandene PERSON_NR wird verwendet!
29.11.2006 17:09:11: Datensatz Nr.10: Neuer Artenfund mit ARTENFUND_NR = 854511 und
                                      LFUNRN = 0101562 und Beobachtungsdatum = 2004 wurde angelegt.
29.11.2006 17:09:11: Datensatz Nr.10: Neue Status_ID = 0 von ARTENFUND_NR = 854511 wurde angelegt.
```

Abbildung 4.3: Ausschnitt aus der Log-Datei

```
29.11.2006 17:47:17: Datensatz Nr.3020: Neue Status_ID = 0 von ARTENFUND_NR = 857183
                     wurde angelegt.
29.11.2006 17:47:18: Datensatz Nr.3020: Es wurde(n) 4 Person(en) neu angelegt.
29.11.2006 17:47:18: Datensatz Nr.3020: Es wurde(n) 2682 Artenfund(e) neu angelegt.
29.11.2006 17:47:18: Datensatz Nr.3020: Es wurde(n) 137 Fundort(e) neu angelegt.
29.11.2006 17:47:19: Datensatz Nr.3020: Es wurde(n) 2682 Artstatus/Artstati neu angelegt.
29.11.2006 17:47:19: Datensatz Nr.3020: Import mit Import Nr 27 wurde durchgeführt.
29.11.2006 17:47:19: Datensatz Nr.3020: ***********************************************
```

Abbildung 4.4: Meldung über erfolgreichen Import in der Log-Datei

5 Konzeption des Internet-Kartendienstes

Vor Erstellung eines Internet-Kartendienstes zur Visualisierung der Artenfunde in Baden-Württemberg müssen die fachlichen Anforderungen, die der Dienst erfüllen muss, analysiert werden. Um abwägen zu können, welches Lösung sich für die Realisierung am besten eignet, stand im Anschluss daran eine Beschäftigung mit den gängigen Map-Server-/Map-Client-Systemen. Diese sollen in Kapitel 5.3 vorgestellt werden.

5.1 Anforderungsanalyse

Die wichtigste Frage für die Darstellung ist, mit welcher Genauigkeit der Fundort einer Art aus Artenschutzgründen dargestellt werden kann und darf. Die Fundortdaten werden aus dem Schema ARTIS entnommen, das zur Zeit die Daten der FlOREIN Kartierung und Daten der § 24a-Biotopkartierung enthält. Als Fundort einer Art wird die Biotopfläche betrachtet, auf der die Art vorkommt. Die Lage des Biotops ist im Schema ALBIS, aus dem die Daten der § 24a-Biotopkartierung übernommen wurden, nur durch den Rechts- und Hochwert des berechneten Flächenmittelpunkts definiert. Da der Artenfundort jedoch in Absprache mit der Fachabteilung nicht punktgenau, sondern nur blattschnittgenau, dargestellt werden soll, kann diese kleine Ungenauigkeit unbeachtet bleiben. Zudem sind die Biotope so kartiert, dass sie an den Grenzen der (TK25-)Kartenblätter enden und auch nur sehr selten und in geringen Flächenanteilen über die Quadrantengrenzen hinausragen. Grund für die nur blattschnittgenaue Darstellung des Artenfundorts ist die Problematik der Rote Liste Arten. Besonders die Fundorte der nach den Kategorien der Roten Liste vom „Aussterben bedrohten“ (Kategorie 1), „stark gefährdeten“ (Kategorie 2) oder „extrem seltenen“ (Kategorie R) Arten dürfen keinesfalls punktgenau dargestellt werden, da ihre Gefährdung durch die Einwirkung des Menschen noch steigen könnte.
In Absprache mit der Fachabteilung soll der Kartendienst so konzeptioniert und erstellt werden, als wäre er von Beginn an der Öffentlichkeit im Internet zugänglich. Daher werden zunächst alle Arten nur blattschnittgenau dargestellt. Dem Wunsch von Referat 24 und 25 nachkommend, soll der Dienst zunächst jedoch nur im Intranet der LUBW zugänglich sein, so dass die Möglichkeit der Kontrolle und Verbesserung der dargestellten Daten durch die Fachleute des Artenschutzes besteht, bevor der Kartendienst der Allgemeinheit präsentiert wird.
Neben der Überlegung, wie der Kartendienst gestaltet werden soll und welche Funktionalitäten er besitzen sollte, muss auch ein möglichst einfacher und übersichtlicher Einstieg in den Dienst gefunden werden.

5.2 Benutzerszenarien

Für die Weiterentwicklung des Kartendienstes können verschiedene Benutzerszenarien in Erwägung gezogen werden. So wäre es z.B. möglich, verschiedene Zugänge einmal für den

interessierten Bürger und für das Fachpublikum (oberste, höhere und untere Naturschutzbehörden, Naturschutzorganisationen) einzurichten. Eine Trennung dieser Zugänge wäre z.B. über einen frei zugänglichen Bereich für alle Gruppen und einen passwortgeschützen, registrierungspflichtigen Bereich für das Fachpublikum möglich. Denkbar wäre auch die Einrichtung zweier verschiedener Dienste im Landes-Intranet und Internet. Allerdings würde so den Naturschutzorganisationen und ehrenamtlichen Naturschützern der Zugang zum erweiterten Angebot verwehrt bleiben.
Abbildung 5.1 zeigt die Startseite des Informationsangebots *Umwelt-Datenbanken und -Karten online*. Dort existiert bereits eine Kategorie *Natur und Landschaft*, in die der Dienst integriert werden könnte.

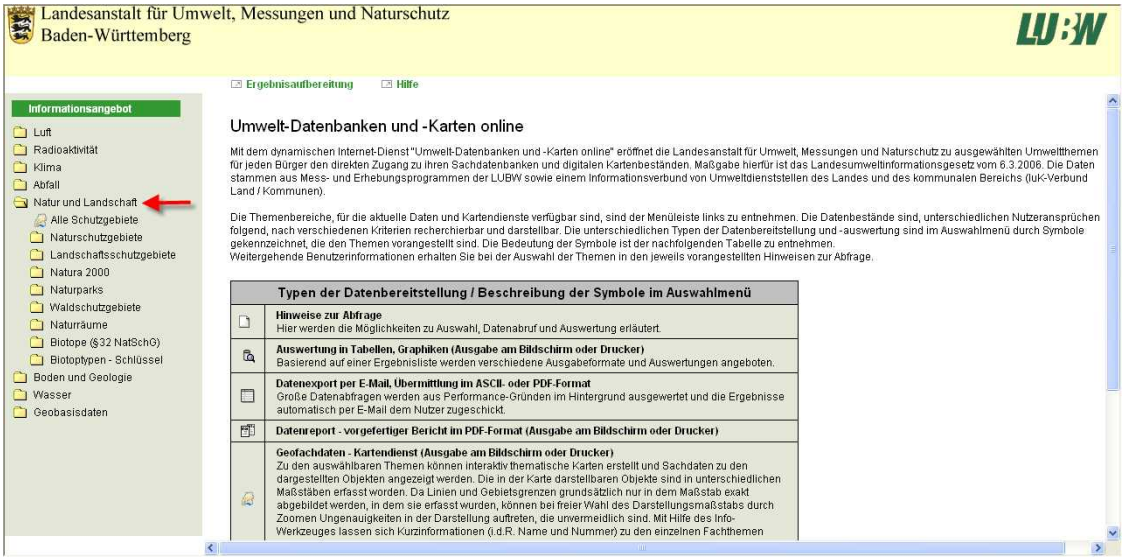


Abbildung 5.1: Das Internetangebot Umwelt-Datenbanken und -Karten online

Welche Fragestellungen im Bezug auf die Artenfundorte könnte es geben, die ein solcher Kartendienst beantworten soll?

Von Seiten der Bürger:

- Wie ist die Verbreitung der Art xy?
- Gibt es in meiner Gemeinde, meinem Landkreis die Art xy?
- Gibt es hier schützenswerte Arten?
- ...

Von Seiten der Behörden:

- Gibt es in unserer Gemeinde Rote Liste Arten? Welche? An welchen Orten?
- Vor Vergabe einer Baugenehmigung: Gibt es auf dem geplanten Bauplatz schützenswerte Arten?
- ...

Einige dieser Fragestellungen kann der Kartendienst, so wie er im Rahmen dieser Arbeit realisiert wurde, noch nicht beantworten. Wird von der Fachseite zu einem späteren Zeitpunkt gewünscht, dass der Kartendienst diese erweiteren Informationen liefert, so muss der Dienst entsprechend angepasst werden. Durch eine Weiterentwicklung der Funktionen, eventuell auch durch Umstieg auf eine andere Technik, wäre dies grundsätzlich möglich.

5.3 Vorstellung verschiedener Map-Server-Systeme

In diesem Kapitel soll eine Auswahl an Map-Server-Systemen, die zur Realisierung des Arten-Kartendienstes dienen könnten, kurz beschrieben werden. Es handelt sich um den Internet Map Server (ArcIMS) der Firma ESRI, die Cadenza Plattform der Firma disy Informationssysteme und den UMN Mapserver, als Vertreter der freien Softwarelösungen.

5.3.1 disy Cadenza

Disy Cadenza ist eine branchen- und betriebssystemunabhängige Plattform, mit der Daten aus verschiedenen Bereichen zusammengeführt und analysiert werden können. Abbildung 5.2 zeigt den Aufbau der Plattform. Grundsätzlich wird zwischen zwei Zugängen unterschieden: Der Desktopanwendung *Cadenza Professional* und der Webanwendung *Cadenza Web*.
Cadenza Professional wird in vielen Umweltbehörden von Baden-Württemberg in unterschiedlichen Ausprägungen, z.B. als UIS-Berichtssystem (BRS) eingesetzt. Die bei der LUBW in Karlruhe zentral vorgehaltene Version des BRS wird über das Landesintranet per Java Web Start aktualisiert.
Cadenza Web ist die in Umfang und Funktionalität gestraffte Version von Cadenza. Diese Anwendung wird im Landesintranet von den Umweltbehörden als BRS-Web und im Internet als öffentlich zugängliche Komponente Umwelt-Datenbanken und -Karten Online (UDO) verwendet. Beide Anwendungen setzen auf einer einheitlichen Plattform auf, die gemeinsame Funktionalitäten bereitstellt (siehe Abbildung 5.2). Der untere Teil der Grafik stellt die Datenbankinfrastruktur dar, auf die die Plattform aufsetzt.
Kern der Plattform ist das „Metadata Repository“. Es ist der zentrale Katalog, in dem in einem offenen XML-Format sogenannte „Informationssichten“ auf die Daten erstellt und gehalten werden. Informationssichten beschreiben sowohl die Eigenschaften der Daten wie auch in welcher Beziehung diese zueinander stehen. In ihnen ist auch festgelegt, welche Informationen dem Benutzer in welcher Weise zur Auswertung angeboten werden.

Der Einstieg in die Recherche ist über die Komponente „Navigator“ möglich. Mit dem sogenannten „Selektor“ lassen sich Kriterien zur Suche nach Informationen in unterschiedlichen Datenquellen bestimmen. Erhält man als Ergebnis der Selektion raumbezogene Daten, so können diese über die in Cadenza integrierte Komponente GISterm, einem Geoinformationssystem mit leistungsfähigen Funktionalitäten, dargestellt werden. Weitere Informationen zu

GISterm und anderen Cadenza-Komponenten sind auf der Webseite der Firma disy unter `http://www.disy.net/disy_gisterm.html` zu finden.
Die Ergebnisse einer Selektion können außerdem in Form von PDF- oder EXCEL-Dokumenten über den Reporter oder den Table Analyzer bzw. in Diagrammen über den Visualizer, ausgegeben werden. Die Anwendung *Cadenza Web Services* soll die Möglichkeit bieten, Daten, die in *CadenzaWeb* bereitgestellt wurden, aus anderen Anwendungen heraus zu nutzen. Dieses Produkt befindet sich zur Zeit noch in der Entwicklungsphase und steht der LUBW noch nicht zur Verfügung. [dis06b]

Abbildung 5.2: Aufbau der disy Cadenza Plattform mit Datenbankinfrastruktur [dis06a]

Über einen in *Cadenza Web* integrierten Map Server können Fach- und Geodaten zusammen in einer Web-Anwendung bereitgestellt, Karten generiert und über eine Webservice-Schnittstelle sogar dynamisch modifiziert werden. Um Karten über das Internet darstellen zu können, bedarf es neben eines Map Servers auch eines Map Clients. Der im Rahmen von Cadenza Web verwendete Map Client basiert auf dem „Community Mapbuilder“, einer OpenSource Entwicklung, die von der OpenSource Geospatial Foundation ins Leben gerufen wurde und von ihr unterstützt wird. Der Map Client kann mit einem einfachen Webbrowser (Mozilla Firefox 1.0+, Internet Explorer 6.0+,...) aufgerufen werden und benötigt keine Ergänzungs- oder Zusatzmodule. Die Kommunikation zwischen Map Server und Map Client basiert auf dem OGC(Open Geospatial Consortium)-WMS(Web Map Service)-Standard. [Ope06]
Zur Erstellung von Karten, die über den Map Client angezeigt werden sollen, wird üblicherweise zunächst eine Basiskarte in GISterm vorkonfiguriert, die dann in einer XML-Datei im MML-Format (Map Markup Language) im sogenannten „Cadenza Repository“ abgespeichert wird. Unter einem Repository versteht man ein Datenhaltungssystem, in dem alle Informationen über wiederverwendbare Softwarebausteine enthalten sind [Uni03]. Beim Aufruf einer Karte lädt der Map Server diese Datei über das GISterm Framework, welches die Karte in

eine Pixelgrafik umwandelt und das Ergebnis der Anfrage an den Server zurücksendet, der es wiederum an den Map Client weiterleitet.
Disy Map Client/Map Server werden im Intranet bereits für einige Kartendienste verwendet. Allerdings befindet sich diese Technologie noch in stetiger Entwicklung.

5.3.2 UMN MapServer

Eine weitere Möglichkeit dynamische Karteninhalte im Internet darzustellen, bietet der UMN MapServer. Im Gegensatz zu den beiden anderen vorgestellten Kartendiensttechniken wird der UMN MapServer an der LUBW bisher noch nicht verwendet. Im Rahmen einer Diplomarbeit wird jedoch derzeit erprobt, ob diese Technik zukünftig für Kartendienste im Internetangebot der LUBW angewendet werden könnte. Daher soll an dieser Stelle ein kurzer Überblick über diese relativ neue Technik gegeben werden.

Der Ursprung dieses Map Servers liegt im „ForNet Projekt“ der University of Minnesota (UMN), die ihn in Zusammenarbeit mit der NASA und dem Minnesota Department of Natural Resources (MNDNR) entwickelt hat. Zur Zeit wird der MapServer vom „TerraSIP Projekt“ beherbergt, einem durch die NASA finanzierten Projekt der UMN und dem Consortium of Land Management Interests.
Bei der UMN MapServer-Software handelt es sich um eine OpenSource-Entwicklungsumgebung zur Erstellung von Anwendungen mit dynamischen Karteninhalten. Die Anwendung ist CGI-basiert und enthält eine OGC-WMS-konforme Schnittstelle. Den UMN MapServer zeichnet besonders seine hohe Stabilität und Geschwindigkeit auch in großen, vernetzten Anwendungen aus. Der größte Vorteil liegt jedoch in der Tatsache, dass hinter diesem OpenSource Projekt eine große, weltweite Anwendergemeinschaft steht, die täglich ihr Wissen in das Projekt einbringt und so die Entwicklung stetig vorantreibt. Der UMN MapServer ist plattformunabhängig und lässt sich für die meisten UNIX/Linux, Microsoft Windows und auch MacOS Betriebssysteme (ab OSX) kompilieren. Das in der Software enthaltene Modul „MapScript“ erlaubt den Zugang zur Application Programming Interface-Schnittstelle (API) über die Skriptsprachen Perl, PHP, Python und Java. Enthalten die Daten eine räumliche Komponente und können diese mit einer der oben genannten Skriptsprachen bearbeitet werden, so sind die Daten auch in einer Karte darstellbar. Über das Perl-DBI-Modul können Geodaten aus Datenbanken (Oracle, MySQL, Sybase etc.) auch zusammen mit traditionellen Kartenformaten (Raster- und Vektorgraphikformate) in eine Webanwendung eingebunden werden. [UMN06]

5.3.3 ESRI Internet Map Server - ArcIMS

ArcIMS der Firma ESRI ist der wohl bekannteste kommerzielle Vertreter der Map-Server-Systeme, die zur Zeit auf dem Markt konkurrieren. Neben der Bereitstellung von Kartendiensten im Intra- und Internet besitzt der ArcIMS viele weitere Funktionen um Geodaten aufzubereiten und verschiedenen Nutzern zugänglich zu machen. Er wurde als mehrschichtiges System entwickelt. Dies bedeutet, dass die einzelnen Funktionen in logische Einheiten zusammengefasst werden können. So lässt sich der IMS in die *Präsentationsschicht* (Presentation Tier), die *Business Logik Schicht* (Business Logic Tier) und die *Daten-Speicherungsschicht* (*Data Storage Tier*) untergliedern. Abbildung 5.3 verdeutlicht dies.

Auf die einzelnen Komponenten der verschiedenen Schichten soll später genauer eingegangen werden.

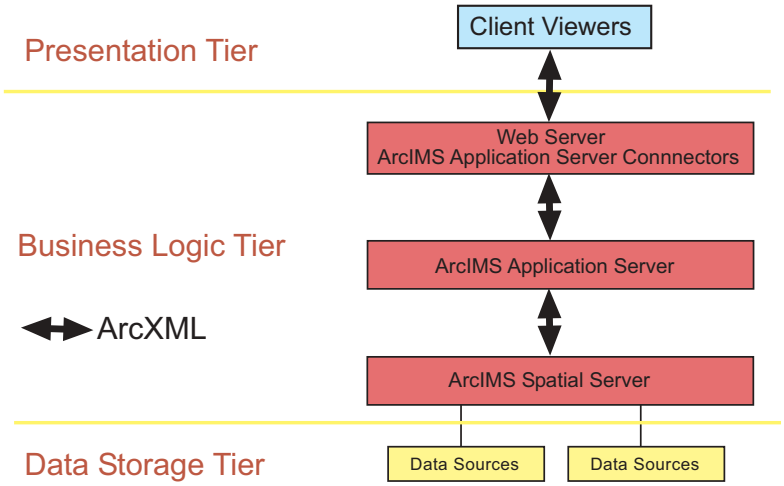


Abbildung 5.3: Die Mehrschichtenarchitektur des ArcIMS (nach: [ESR04a])

Zur Verwendung von ArcIMS bedarf es zusätzlicher Komponenten (siehe Abbildung 5.4), die nicht im Softwarepaket mitgeliefert werden. Ein Webserver, z.B. Apache oder IIS, regelt allgemein Anfragen eines Clients und schickt die Antwort an ihn zurück. Des Weiteren wird noch ein sogenanntes „Servlet Engine“ benötigt, das eine Erweiterung des Webservers darstellt und das Bindeglied zwischen der JavaVM und dem Webserver bildet. An der LUBW wird der IIS in Kombination mit dem *Tomcat Servlet* verwendet. Eine JavaVM (Virtual Machine) stellt die Programmierschnittstelle (API) bereit, die viele java-basierte Komponenten des IMS benötigen. Die JavaVM ist in einer Java Runtime Environment (JRE) oder im Java Development Kit (JDK) enthalten.

Um über das Internet auf einen ArcIMS Dienst zuzugreifen, können verschiedene Clients eingesetzt werden. Von ArcIMS werden auf Clientseite (*Presentation Tier*) leistungsfähige HTML- und Java-Viewer zur Verfügung gestellt. Am häufigsten wird der HTML/Javascript-Viewer verwendet.

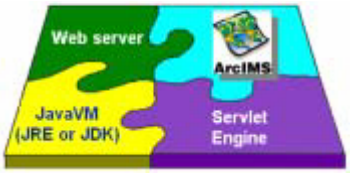


Abbildung 5.4: Zusatzkomponenten zu ArcIMS [ESR04a]

Es besteht auch die Möglichkeit, über Desktop Produkte wie ArcGIS oder drahtlose Geräte wie WAP-Handys oder PDAs über vorbereitete Schnittstellen auf die ArcIMS-Dienste zuzugreifen. Als Systemumgebung für den Client eignet sich ein einfacher Web-Browser. Der Kauf oder Download spezieller Zusatzsoftware erübrigt sich somit für den Nutzer. Abhängig

von der Art und Konfiguration des Viewers kann sein Funktionsumfang von der einfachen Kartenanzeige bis zu komplexen GIS-Funktionen reichen.

Auf Serverseite befindet sich die sogenannte *Business Logic Tier*. Diese setzt sich aus den Komponenten *Webserver* inklusive *Application Server Connectors*, *Application Server* und *Spatial Server* zusammen. Diese Komponenten werden benötigt, um einen Dienst zu betreiben und um Anfragen der Clients zu bearbeiten. Innerhalb dieser Schicht wird über die ESRI-eigene Sprache ArcXML kommuniziert, die sich stark an XML anlehnt. Abbildung 5.5 verdeutlicht den Aufbau dieser Schicht.

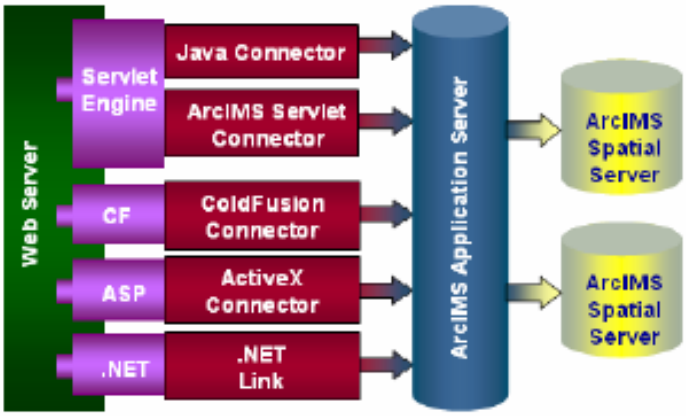


Abbildung 5.5: Komponenten der Business Logic Tier [ESR04a]

Als Vermittler zwischen dem *Client* und dem *Application Server* dient der *Webserver*. Bei der LUBW wird als Webserver ein Microsoft Windows Server verwendet, der die Microsoft Internet Information Services (IIS) zur Verfügung stellt.
Erhält der Webserver eine Anfrage von einem Client, so leitet er diese über das *Servlet Engine* und einen passenden *Connector* (*ActiveX Connector, ColdFusion Connector oder ArcIMS Servlet Connector*) an den *Application Server* weiter, der wiederum die Anfrage für den *Spatial Server* aufbereitet. Die Konnektoren werden benötigt um Anfragen, die nicht in ArcXML geschrieben sind, für den *Application Server* in ArcXML zu übersetzen und umgekehrt auch Antworten des Servers wieder in HTML umzuwandeln.
Der *ColdFusion Connector* wird für die Übersetzung von Anfragen, die von ColdFusion Seiten kommen, benötigt. ColdFusion ist eine Technologie, die zur Erstellung von Web-Applikationen verwendet wird. Sie steht damit in Konkurrenz zu anderen serverseitigen Technologien wie Perl,PHP oder ASP.
Der *ActiveX Connector* übersetzt Anfragen von ActiveServer Pages oder ActiveX-Anwendungen, die in VB, Delphi oder C++ geschrieben sind. Am häufigsten wird der *ArcIMS Servlet Connector* verwendet, der auf Basis eines Java Servlets läuft. Web Client und Webserver kommunizieren miteinander über das Hypertext Transfer Protocol (HTTP).

Die tatsächliche Bearbeitung der Client-Anfragen geschieht auf dem *Spatial Server*. Auf ihm sind alle ArcIMS Dienste abgelegt. Er ist als Ablaufcontainer für verschiedene Serverkomponenten zu sehen, die er zur Aufbereitung von Client-Anfragen benötigt. Diese Komponenten, die auch als Servertypen bezeichnet werden, sind in Abbildung 5.6 dargestellt.

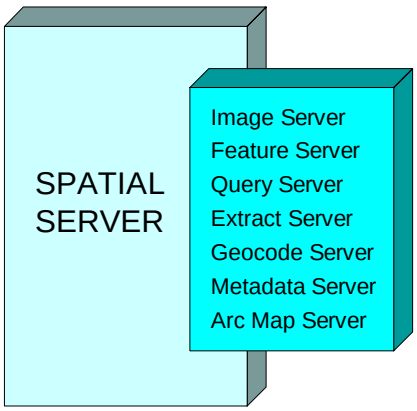


Abbildung 5.6: Komponenten des Spatial Server

Je nachdem um was für eine Art von Anfrage es sich handelt, verwendet der *Spatial Server* zur Bearbeitung der Anfrage eine der rechts aufgeführten Serverkomponenten.
Am häufigsten finden Image Dienste, die vom *Image Server* verarbeitet werden und ArcMap Image Dienste, die vom *ArcMap Server* verarbeitet werden, Verwendung. Bei diesen Diensten handelt es sich um die eigentlichen Kartendienste. Ein *Image Server* generiert bei jeder neuen Serveranfrage eine Karte und sendet diese als JPEG- (Joint Photographic Experts Group), PNG- (Portable Network Graphics) oder GIF- (Graphics Interchange Format) Bild an den Client zurück. Die Vorgaben für die Erzeugung der Karte sind in der Konfigurationsdatei (axl-Datei) festgeschrieben (siehe auch Kapitel 6.1).
Im Gegensatz zum *Image Server* generiert der *ArcMap Server* die Karten auf Basis eines ArcMap-Dokuments. Dieses wird in der ArcGIS-Komponente ArcMap mit allen benötigten Layern angelegt. Die Karte wird bereits in ArcMap graphisch so gestaltet, wie sie später im Kartendienst angezeigt werden soll. Auf diese Weise kann nahezu alles, was in ArcMap darstellbar ist, über den *ArcMap Server* im Internet präsentiert werden. Die nachfolgend beschriebenen Servertypen werden bei der LUBW nicht verwendet, sollen jedoch aus Gründen der Vollständigkeit ebenfalls angesprochen werden.

Der *Feature Server* erlaubt, Vektordaten an den Client zu senden (Feature Streaming), mit denen der Benutzer dann eine Karte individuell gestalten und komplexe Aufgaben wie z.B. eine erweiterte Beschriftung realisieren kann. Allerdings können die Feature Streaming Funktionen nur von „High-End-Clients“ wie dem ArcIMS Java Viewer, ArcExplorer und ArcMap genutzt werden.
Der *Query Server* dient dazu, neue oder vorgefertigte Anfragen auf Daten, die von einem Client an den *Spatial Server* gesendet werden, zu verarbeiten und die Ergebnisse an den Client zurückzusenden. Werden geographische Daten vom Client angefragt, so wird vom *Extract Server* ein Shapefile erstellt, dieses in ein ZIP-Archiv gepackt und an den Client zurückgesendet.
Mit der Geocodierungsfunktion des *Geocode Servers* lassen sich Adressen lokalisieren. Falls vorhanden, sendet der Server auf Anfrage eines Clients die gewünschten exakten geographischen Daten zurück.
Der *Metadata Server* bietet Funktionalitäten, welche die Abfrage und Speicherung von Metadaten über Karten und Geodaten ermöglichen.

Als letzte Ebene der Dreischichtenarchitektur von ArcIMS soll nun noch auf die *Daten-Speicherungsschicht* (*Data Storage Tier*) eingegangen werden. Wie in Abbildung 5.3 erkennbar, bildet sie die Basis dieser Architektur und enthält die Datenquellen, auf die der Dienst später zugreift. Je nach Art des Dienstes können verschiedene Datenformate genutzt werden, die in Tabelle 5.1 augelistet werden.

Datentyp	Datenformat	Image	Feature	ArcMap
Shapefile	Shapefiles	Yes	Yes	Yes
Geodatabase	Geodatabases	No	No	Yes
Personal Geodatabase	Personal Geodatabases	No	No	Yes
Coverages	ArcInfo™Coverages	No	No	Yes
	PC ARC/INFO®Coverages	No	No	Yes
	ArcSDE for Coverages	Yes	Yes	Yes
ArcSDE •SQL Server™ •Informix® •DB2® •Oracle®	ArcSDE Features	Yes	Yes	Yes
	ArcSDE—Versioned Layers	No	No	Yes
	ArcSDE Multiraster and 32-Bit Raster (Oracle)	Yes	No	Yes
	ArcSDE Raster (SQL Server, Informix, DB2)	Yes	No	Yes
CAD	DWG	No	No	Yes
	DXF	No	No	Yes
	DGN	No	No	Yes
Raster	ArcView®Image Catalog	Yes	No	Yes
	ArcSDE Embedded Raster Catalog	Yes	No	Yes
	Personal Geodatabase Managed Catalog	No	No	Yes
	Personal Geodatabase Unmanaged Catalog	No	No	Yes
	ADRG Image (.IMG)	Yes	No	Yes
	ADRG Overview (.OVR)	Yes	No	Yes
	ADRG Legend (.LGG)	Yes	No	Yes
	Band Interleaved by Line (.BIL)	Yes	No	Yes
	Band Interleaved by Pixel (.BIP)	Yes	No	Yes
	Band Sequential (.BSQ)	Yes	No	Yes
	Bitmap—Windows (.BMP)	Yes	No	Yes
	Controlled Image Base (.CIB)	Yes	No	Yes
	CADRG (.CRG)	Yes	No	Yes
	DIGEST ARC Standardized Raster Product (ASRP)	Yes	No	No
	DIGEST UTM/UPS Standardized Raster Product (USRP)	Yes	No	No
	DTED Levels 0, 1, 2 (.DT*)	No	No	Yes
	ERDAS®Imagine (.IMG)	Yes	No	Yes
	ERDAS 7.5 Lan (.LAN)	Yes	No	Yes
	ERDAS 7.5 GIS (.GIS)	Yes	No	Yes
	ERDAS Raw (.RAW)	No	No	Yes
	ER Mapper (.ERS)	No	No	Yes
	ESRI GRID	Yes	No	Yes
	ESRI GRID Stack	No	No	Yes
	Graphic Interchange Format, GIF (.GIF)	Yes	No	Yes
	Impell Bitmap (IMPELL)	Yes	No	Yes
	Intergraph Raster (.CIT, .COT)	No	No	Yes

Datentyp	Datenformat	Image	Feature	ArcMap
Raster (Fortsetzung)	JPEG (.JPG)	Yes	No	Yes
	JPEG 2000 (.JP2)	No	No	Yes
	MrSID®—LizardTech (.sid)	Yes	No	Yes
	MrSID Gen 3 (.sid)	No	No	Yes
	National Image Transfer Format (.NTF)	Yes	No	Yes
	Portable Network Graphics (.PNG)	No	No	Yes
	SunRaster File (SUN)	Yes	No	No
	Tagged Image File Format (.TIF)	Yes	No	Yes
	TIFF with Geo Header (.TIF)	Yes	No	Yes
Other	Annotation Layers	No	No	Yes
	TIN	No	No	Yes
	VPF	No	No	Yes
	Text files	No	No	Yes
	OLE DB tables	No	No	Yes
	SDC	Yes*	No	Yes

Tabelle 5.1: Mögliche Datenformate zur Eingabe in ArcIMS ([ESR04a])

5.4 Fazit zum Thema Map-Server-Systeme

Sowohl auf Client-, wie auch auf Serverseite gibt es inzwischen eine Vielzahl von Technologien und Softwareprodukten, die zur Erstellung und Präsention eines Kartendienstes für das Internet zur Verfügung stehen. Jede dieser Lösungen besitzt Vor- und Nachteile, die im Hinblick auf das Projekt gegeneinander abgewogen werden müssen.
Größter Vorteil der OpenSource-Produkte, wie dem UMN MapServer – in Kombination mit freien Clientlösungen – ist sicherlich die Tatsache, dass sie kostenlos zur Verfügung stehen. Allerdings kann nicht vorausgesagt werden, ob und wie diese Komponenten in bereits bestehende Systemumgebungen integriert werden können. Stellt sich im Rahmen der in Kapitel 5.3.2 angesprochenen Erprobung des UMN MapServers heraus, dass sich diese Lösung für die Kartendienste der LUBW eignet, wäre eine Migration auf dieses System zukünftig vorstellbar.

Ein Vorteil der Lösung der Firma disy Informationssysteme auf Basis der *Cadenza Plattform* ist, dass viele Geodaten bereits über disy Cadenza vorkonfiguriert sind und so die Migration von bereits bestehenden Diensten auf dieses System ermöglichen. Allerdings erfordert die Einrichtung der Repositories und die Erstellung der Selektoren, mit denen Daten vorselektiert werden können, eine umfangreiche Einarbeitung in den Aufbau dieser Lösung und einen erheblichen Arbeitsaufwand bei der Erstellung, da bislang kaum Anleitungen dazu vorhanden sind. Ist das Know-How jedoch vorhanden, kann auch ein bestehender Dienst auf diese Technologie umgestellt werden.

Letztendlich fiel die Entscheidung für eine Implementierung des Kartendienstes mit ArcIMS. Die Kombination von HTML Viewer und *Image Server* bietet alle Funktionalitäten, die für den Kartendienst benötigt werden. Der Vorteil des HTML Viewers gegenüber dem Java Viewer ist, dass er lediglich einen Browser zur Ausführung benötigt und der Benutzer keine weiteren Komponenten installieren muss.

Da bereits zahlreiche Kartendienste auf Basis von ArcIMS im Internet und Landesintranet realisiert sind, ist die serverseitige Systemumgebung bereits auf diese Lösung eingerichtet und ein umfassendes Fachwissen im ITZ vorhanden. Zudem ermöglicht die ArcIMS-Software durch eine Vielzahl von Werkzeugen und Assistenten eine zügige Implementierung eines neuen Dienstes. Treten dennoch Probleme bei der Entwicklung des Kartendienstes auf, erhält man auf den Webseiten der Firma ESRI einen guten Support (User Foren, Whitepapers etc.).

6 Implementierung des Kartendienstes mit ArcIMS

Der Aufbau eines Kartendienstes mit ArcIMS erfordert mehrere Schritte. Der genaue Ablauf, der zur Erstellung des Dienstes notwendig ist, kann Abbildung 6.1 entnommen werden. Erklärungen zu den einzelnen Schritten finden sich in den nachfolgenden Unterkapiteln.
Zur Umsetzung der Kartendienst-Anwendung zählt neben der Implementierung des eigentlichen Kartendienstes mit ArcIMS auch die Erstellung einer Einstiegs-Internetseite und weiterer Seiten, die wertvolle ergänzende Informationen über Arten liefern. Die hinter diesen Seiten stehende Technik soll in diesem Kapitel ebenfalls erläutert werden.

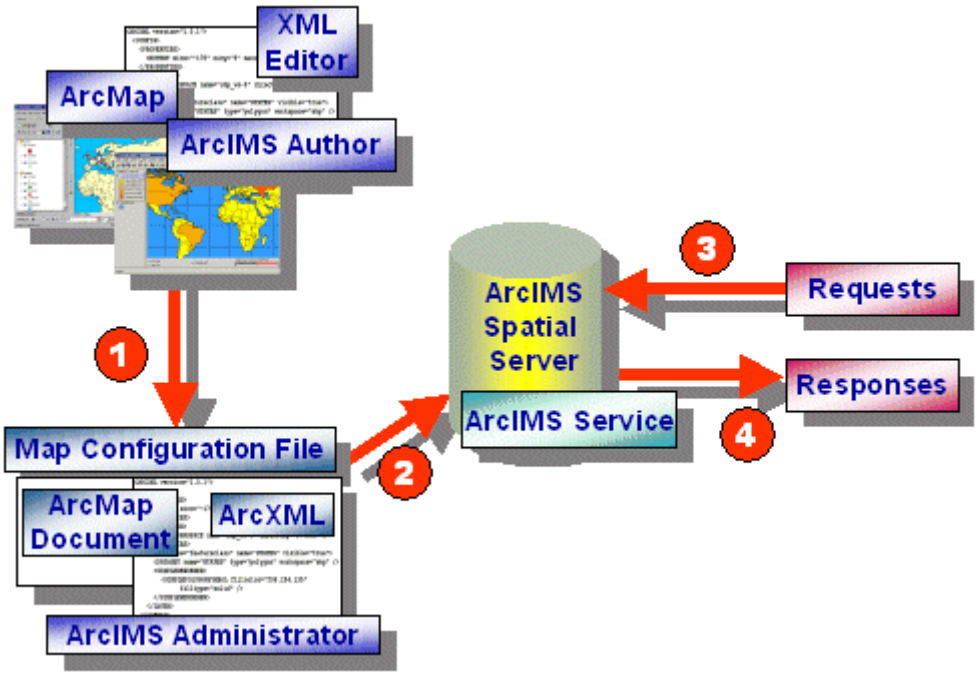


Abbildung 6.1: Ablauf der Erstellung eines ArcIMS Service [ESR04b]

6.1 Erstellen einer Konfigurationsdatei (*.axl Datei)

Der erste Schritt zum Aufbau eines Kartendienstes mit ArcIMS ist die Erstellung einer Konfigurationsdatei. Diese kann entweder über das Werkzeug „Author“ angelegt werden oder es werden alle benötigten Angaben direkt in einem XML Editor in der ArcGIS-spezifischen Sprache ArcXML in eine Datei geschrieben. Diese wird abschließend mit der Endung .axl

abgespeichert. In der Konfigurationsdatei wird festgelegt, welche Layer in der Karte angezeigt, in welchem Maßstabsbereich diese sichtbar und wie sie farblich gestaltet sein sollen. Die Reihenfolge der Layer in der Karte ist durch Reihenfolge der Nennung in der axl-Datei festgelegt. Dabei ist der in der Datei zuerst genannte Layer derjenige, der beim Aufbau der Karte zuerst gezeichnet wird und daher ganz unten liegt.
In dieser Datei findet sich auch die Angabe, wo die Datenquellen abgelegt sind. Als Datenquellen können alle in Tabelle 5.1 aufgeführten Datenformate verwendet werden. Die axl-Datei dient als Input für die Anlage des Image Service. Der vollständige Quellcode dieser Datei befindet sich im Anhang dieses Dokuments. Nachfolgend ein Ausschnitt aus der axl-Datei:

```
<?xml version="1.0" encoding="UTF-8"?>
<ARCXML version="1.1">
  <CONFIG>

<ENVIRONMENT>
      <LOCALE country="DE" language="de" variant="" />
      <UIFONT color="0,0,0" name="SansSerif" size="12"
      style="regular" />
      <SCREEN dpi="96" />
</ENVIRONMENT>

<MAP>
<PROPERTIES>
  <ENVELOPE minx="3373732,9699999997" miny="5254209,109229076"
   maxx="3676781,525" maxy="5532892,065881057"
   name="Initial_Extent" />
   <MAPUNITS units="meters" />
   <FEATURECOORDSYS id="31467" />
   <FILTERCOORDSYS id="31467" />
   <BACKGROUND color="255,255,255" transcolor="255,255,255"/>
</PROPERTIES>

<WORKSPACES>
    <SHAPEWORKSPACE directory="\\itzgs2\fix_data\bw_uebersicht"
                    name="shp_ws-100"/>
    <SHAPEWORKSPACE directory="\\itzgs2\fix_data\artenfundort"
                    name="shp_ws-101"/>
    <SDEWORKSPACE name="sde_ws-0" server="ripssde"
    instance="port:5151" database="" user="lfu_webview"
    'encrypted="true" password="GEOPGWDQUCUNWX"/>
</WORKSPACES>

<LAYER type="image" name="TK 25" id="1" visible="true"
  minscale="1:5000" maxscale="1:12500">
  <DATASET workspace="sde_ws-0"
   name="ADMIN.UIS_0100000005800001.RASTER"/>
  <IMAGEPROPERTIES transparency="1" />
     <COORDSYS id="31467" />
</LAYER>

<LAYER type="featureclass" name="Blattschnitt TK 25 Quadranten"
       visible="true" id="tk25bs_q">
```

```
  <DATASET name="tk25bs_q2" type="polygon"
   workspace="shp_ws-101" />
  <SIMPLERENDERER>
   <SIMPLEPOLYGONSYMBOL filltype="solid" filltransparency="0,5"
    fillcolor="221,255,217" boundarywidth="1"'
    boundarycolor="255,153,153"/>
  </SIMPLERENDERER>
        <COORDSYS id="31467" />
</LAYER>
```

Tabelle 6.1 enthält die Layer, in der Reihenfolge, wie sie im Kartendienst angeordnet sind. Der Layer mit der Nr. 0 ist der oberste Kartenlayer.

Layer_Nr	Name des Layers	Sichtbar im Maßstabsbereich	Sichtbar beim Start
0	Verwaltungssitz	1:400.000 bis 1: 5.000.000	nein
1	Biotop (gesamt)	1:1 bis 1: 30.000	nein
2	Biotope (Auswahl >10ha)	1:400.000 bis 1:5.000.000	nein
3	Biotope (Auswahl > 5ha)	1:150.000 bis 1: 400.000	nein
4	Biotope (Auswahl > 2ha)	1: 50.000 bis 1: 150.000	nein
5	Biotope (Auswahl > 1ha)	1: 30.000 bis 1: 50.000	nein
6	Kreis	1:1 bis 1:5.000.000	ja
7	Blattschnitt TK 25 Quadranten	1:1 bis 1:5.000.000	ja
8	Bodensee	1:400.000 bis 1:5.000.000	ja
9	ÜK 500	1:125.000 bis 1: 300.000	nein
10	TÜK 200	1:50.000 bis 1:125.000	nein
11	TK 100	1: 25.000 bis 1: 50.000	nein
12	TK 50	1: 12.500 bis 1: 25.000	nein
13	TK 25	1: 5.000 bis 1:12.500	nein

Tabelle 6.1: Die Layer des ArcIMS Kartendienstes

6.2 Anlegen und Starten eines Image Service über den ArcIMS Administrator

Der ArcIMS Administrator ist das Werkzeug zur Verwaltung der ArcIMS Dienste. Der Benutzer kann über ihn Dienste hinzufügen, löschen, starten und stoppen.
Abbildung 6.2 zeigt die Eingabemaske zur Erstellung des Image Service. Unter „Map File“ wird der Speichertort der zuvor erstellte axl-Datei angegeben.
Der über den ArcIMS Administrator angelegte Dienst wird auf einem *Spatial Server* gespeichert. In diesem Fall handelt es sich um den Image Server1, der als Teil eines *virtuellen Servers* verstanden werden kann. In einem *virtuellen Server* werden mehrere *Spatial Server* zusammengefasst.
Erhält der *Spatial Server* eine Anfrage, wie z.B. *getImage*, von einem Client (hier wird der HTML Client verwendet), so verarbeitet er diese und gibt als Ergebnis eine generierte Karte zurück. Im unteren Bereich der Maske, im Register „Server Output“, muss angegeben werden, in welchem Format die Karte zurückgeliefert werden soll. Hier wurde unter `Image Type`

angegeben, dass vom Server ein Bild im Format „Portable Network Graphics 8-Bit“ (PNG) erzeugt werden soll. Außerdem muss der Ort des „Output-Verzeichnisses“ auf dem Server angegeben werden. In diesem werden alle vom *Spatial Server* erzeugten Kartenbilder abgelegt. Auf dieses Verzeichnis kann der Client dann via HTTP zugreifen.

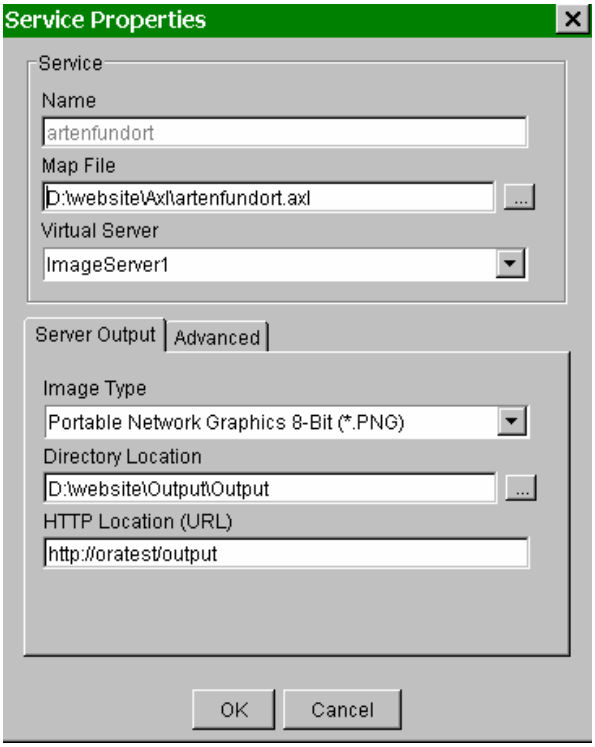


Abbildung 6.2: Maske zur Einrichtung eines ArcIMS Service

6.3 Erstellung der Viewer-Webseite

Die Erstellung der Webseite für den Viewer kann über den ArcIMS Designer erfolgen. Er erlaubt die Auswahl des Viewertyps (HTML oder Java Viewer) und die Konfiguration des Aussehens der Webseite und der gewünschten Funktionalitäten über einen einfachen Assistenten (Wizard). Dieser erstellt automatisch ein Verzeichnis mit dem Namen des Dienstes. In diesem befinden sich alle vom Viewer benötigten HTML- und Javascript-Dateien. Abbildung 6.3 zeigt, wie der Viewer nach der Erstellung mit dem Wizard aussieht.
Im linken Seitenbereich (Frame) befindet sich die Werkzeugleiste (Toolbar), im mittleren Frame die Karte mit der Übersichtskarte, dem Nordpfeil und der Maßstabsleiste und im rechten Frame die Layerliste, in der Layer ein- und ausgeblendet und aktiv geschaltet werden können.
Deutlich zu erkennen ist, dass sich in der Toolbar sehr viele Werkzeuge befinden, die für einen einfachen Kartendienst nicht unbedingt benötigt werden und die Handhabbarkeit unnötig erschweren. Unter anderem aus diesem Grund wurde an der LUBW ein Viewer entwickelt, der eine in seinen Funktionalitäten abgespeckte Version des „ESRI Standard HTML Viewers“ darstellt. Durch die Löschung nicht benötigter Dateien konnte der Speicherplatz, den der

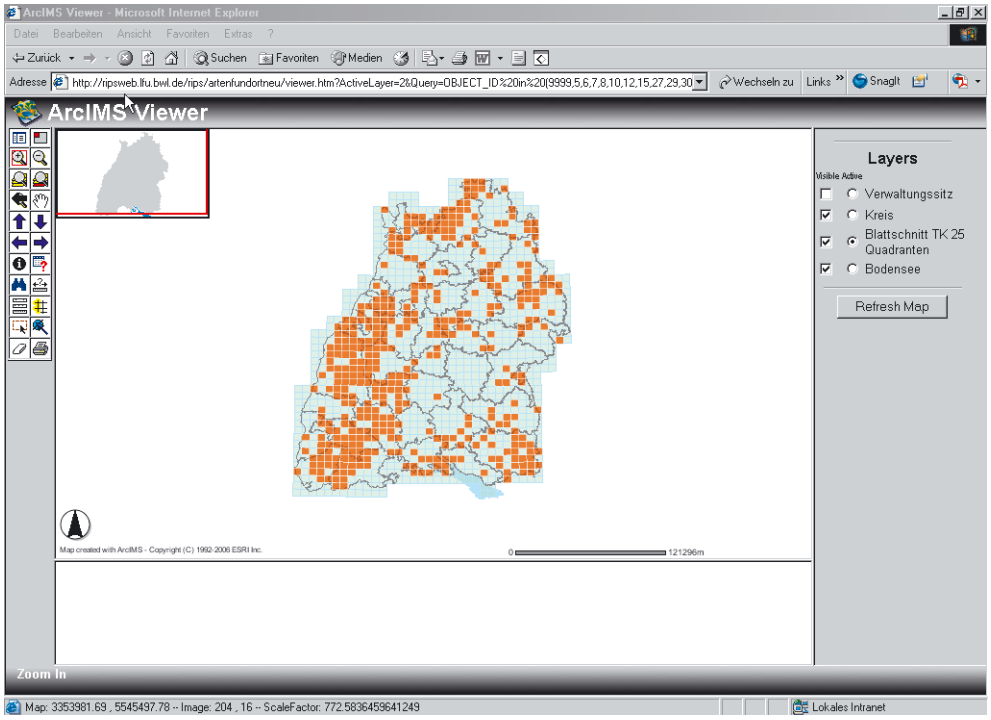


Abbildung 6.3: Der Standard HTML Viewer der Firma ESRI

Viewer benötigt, auf fast 1/3 der ursprünglichen Menge verringert werden. Um die Bedienbarkeit zu verbessern, werden dem Nutzer zudem nur die wirklich notwendigen Werkzeuge zur Verfügung gestellt. Ein weiterer Grund für die Veränderung des Standard Viewers liegt in den Layoutvorgaben der LUBW. Alle Kartendienste, die im Bereich Umwelt-Datenbanken und -Karten online angeboten werden, sollen möglichst das gleiche Layout besitzen. Änderungen am Viewer lassen sich generell durch Veränderungen im Quellcode der verschiedenen HTML- und Javascript-Dateien erreichen. In HTML-Dateien wird in erster Linie das Aussehen der Webseite beschrieben. Sie definieren also den Inhalt der verschiedenen Frames, wohingegen die Javascript-Dateien Funktionen für die Interaktion des Benutzers mit der Karte bereitstellen. Die Datei `ArcIMSparam.js` muss modifiziert werden, wenn Änderungen am Viewer selbst vorgenommen werden sollen. Hier sind die Variablen zu finden, die sowohl das Aussehen wie auch das Verhalten der Webseite beeinflussen. In dieser Datei kann durch Veränderung der Variablen z.B. die Werkzeugleiste verändert, die Ausdehnung der Karte eingestellt und festgelegt werden, welche Informationen bei einer Attributabfrage mit dem „Identify“ Werkzeug ausgeben werden sollen.
Im von der LUBW verwendeten Viewer sind deutlich die Veränderungen gegenüber dem Standard HTML Viewer der Firma ESRI zu erkennen (vergleiche dazu Abbildung 6.3 und 6.4).
Auch für den im Rahmen der Diplomarbeit erstellten Kartendienst war die Verwendung des LUBW-Templates (Vorlage) geplant. Leider hat sich bei der Erprobung des Dienstes herausgestellt, dass dem modifizierten Template eine oder mehrere Funktionen bzw. Dateien fehlen, die die farbige Hervorhebung der Quadranten, in denen die gewählte Art vorkommt, ermöglichen würden. Aus diesem Grund wird wurde der Dienst daher mit Standard-ESRI-Template

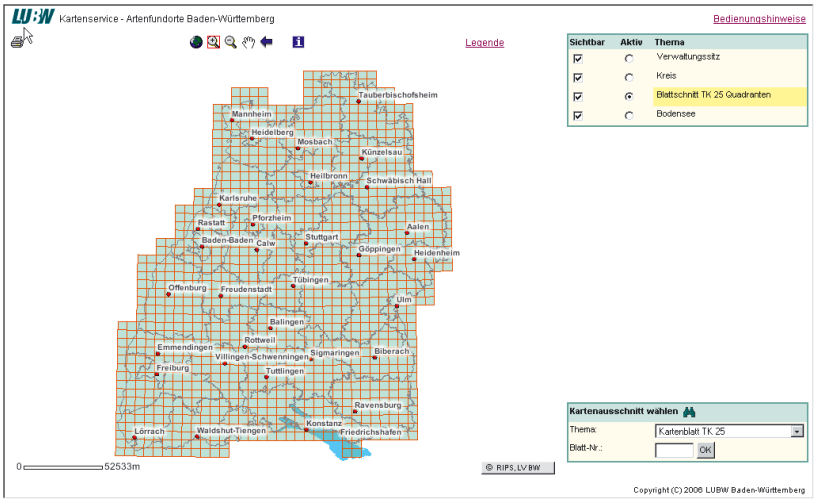


Abbildung 6.4: Der Kartendienst im LUBW Layout

angelegt, bis im ITZ ein neues LUBW-Template erstellt ist, das diese Funktionalität bietet.

6.4 Der Einstieg in den Kartendienst

Um den Einstieg in den Kartendienst möglichst komfortabel zu gestalten, wurde eine Webseite angelegt, auf welcher der Anwender über Auswahllisten eine Art auswählen kann. Würden alle Tier- und Pflanzenarten der § 24a-Biotopkartierung in einer Liste ohne vorherige Selektion ausgegeben, würde die Auswahl der gesuchten Art zu lange dauern. Daher wird der Suche nach Arten eine Einschränkung nach Artengruppen vorangestellt. Es besteht die Möglichkeit, die Auswahl über deutsche oder wissenschaftliche Artnamen zu treffen. Aus Übersichtsgründen wurden die Auswahlboxen nach deutschen bzw. wissenschaftlichen Artnamen auf verschiedenen Webseiten angelegt, die jedoch untereinander verlinkt sind und so einen einfachen Wechsel zwischen den beiden Suchmöglichkeiten gestatten. Abbildung 6.5 zeigt die Webseite, welche die Auswahl nach deutschen Artnamen erlaubt.
Die Inhalte der Auswahllisten werden, für den Nutzer unsichtbar, über ein PHP-Skript (artensuche.php), welches über das Oracle Call Interface (OCI) auf die Oracle Datenbank zugreift, abgerufen. Das OCI ist eine Schnittstelle, die Funktionen für den komfortablen Zugriff auf Oracle Datenbanken bereitstellt. Um diese verwenden zu können, müssen auf dem *Webserver* das Oracle8 CallInterface (OCI8) und verschiedene Oracle8-Client-Bibliotheken (Libraries) installiert sein.

Wird eine bestimmte Artengruppe in der ersten Liste ausgewählt, so werden dynamisch die zu dieser Gruppe gehörenden Namen der Tier- und Pflanzenarten aus der Datenbank abgefragt und in der zweiten Liste angeboten. Durch eine Verknüpfung der Artenliste des Artenlexikons mit den Tabellen des Schema ARTIS, werden in der zweiten Auswahlbox immer nur die Namen der Arten angezeigt, die auch im Schema ARTIS vorhanden sind.
Bestätigt der Anwender seine Auswahl am Ende der Seite mit OK, wird die zur Art gehörende eindeutige Artnummer an ein weiteres PHP-Skript (artenfundort.php) übergeben, und

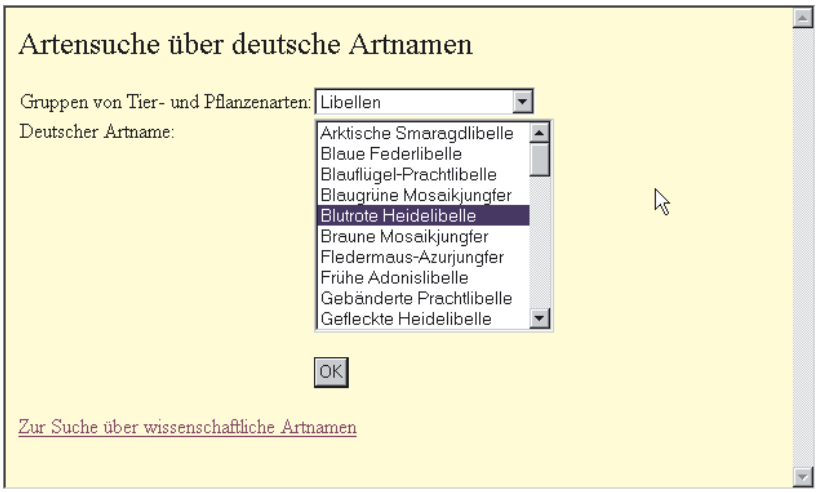


Abbildung 6.5: Die Internetseite "Artensuche über deutsche Artnamen"

die HTML-Seite „Artenfundort“ (siehe Abbildung 6.6) wird in einem neuen Browserfenster geöffnet. Auf dieser bekommt der Benutzer dann die eindeutige Artnummer, die deutsche und wissenschaftliche Artbezeichnung und die Gefährdungskategorie angezeigt. Selbstverständlich können zur jeweiligen Art auch weitere Informationen aus der Datenbank ausgegeben werden.

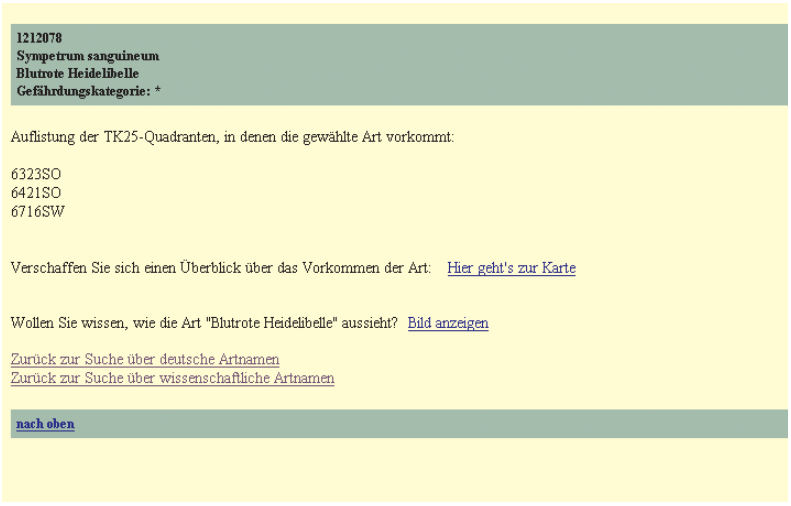


Abbildung 6.6: Die Internetseite "Artenfundort"

Hierzu muss zunächst die folgende SQL-Abfrage um den Namen der Spalte, welche die gewünschten Informationen enthält, erweitert werden.

```
SELECT LFUNRN, WISSNAME, DTNAME FROM B24AVN_ARTEN WHERE LFUNRN=
 '$art_lfunrn';
```

Außerdem muss noch eine weitere Codezeile in das Skript eingefügt werden, in der anstelle von `DTNAME` der Name der Tabellenspalte eingetragen wird. Der Quellcode lautet dann wie

folgt:

```
<?echo OCI_Result ($resultART, "DTNAME")?><br>
```

Neben den Informationen im grünen Balken erhält man hier auch eine Liste der TK-Quadranten, in denen die gesuchte Art vorkommt.
Ausgehend von dieser Seite kann der Benutzer sich dann eine Karte anzeigen lassen, in der die Verbreitung der gewählten Art dargestellt wird, oder sich Bilder zu der Art anzeigen lassen, sofern diese im Bildarchiv vorhanden sind. Das Naturschutzbildarchiv ist bereits Teil der LUBW-Anwendung NafaWeb. Es enthält neben den Bildern zu Tier- und Pflanzenarten auch Fotos zu weiteren Themengebieten. Durch die Einbindung des Bildarchivs in die Kartendienst-Anwendung werden dem Benutzer wertvolle ergänzende Informationen in Form von Bildern geliefert.

Abbildung 6.7 zeigt ein Beispiel für die Anzeige einer Seite der Bilddatenbank.

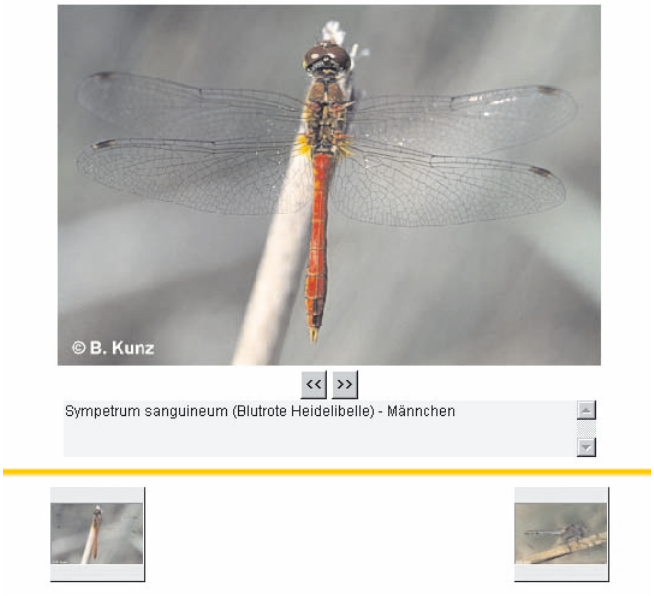


Abbildung 6.7: Beispiel für ein Bild aus dem Bildarchiv

6.5 Darstellung der Artenfundorte im Kartendienst

Die Darstellung der Artenfundorte im HTML Viewer erfolgt durch die farbige Hervorhebung der Quadranten, in denen die gewählte Art vorkommt (siehe Abbildung 6.3). An dieser Stelle soll erläutert werden, wie diese Darstellung technisch realisiert wird. Aus diesem Grund blicken wir nochmals auf die bereits angesprochenen PHP-Dateien zurück.
Der Aufruf des HTML Viewers erfolgt auf der Seite artenfundort.php über den Link „Hier geht's zur Karte“. Gibt man lediglich die Adresse des Viewers im Link an, wird der Kartendienst in einem neuen Fenster geöffnet und die Karte von Baden-Württemberg mit dem Layer „Blattschnitt TK 25-Quadranten“ als oberstem Layer angezeigt (ähnlich der Darstellung in

Abbildung 6.4). Gewünscht ist jedoch, dass bereits beim Aufruf des Viewers die Artenfundorte in der Karte markiert sind, die in der vorangegangenen Anfrage ermittelt wurden. Um dies zu realisieren, wird die so genannte „Start-Up URL-Methode“ verwendet. Diese wird vom ArcIMS HTML Viewer bereitgestellt und ermöglicht es, der URL (Uniform Resource Locator) beim Aufruf verschiedene Parameter zu übergeben. Die am häufigsten verwendeten werden nachfolgend aufgelistet:

- **OVMap = OVMapService**
  Ersetzt die Übersichtskarte, die in der ArcIMSparam.js Datei voreingestellt wurde.
- **Box = MinX:MinY:MaxX:MaxY**
  Die Ausdehnung der Karte beim Start.
- **Layers = 0101101**
  Sichtbare Ebenen, beginnend bei der obersten Ebene. 0 steht für „unsichtbar“, 1 für „sichtbar“.
- **ActiveLayer = layerIndex**
  Index der aktiven Ebene.
- **Query = queryExpression**
  Abfrage, die beim Laden des Viewers übergeben werden soll.
- **QueryZoom = Yes**
  Zoom auf das Ergebnis der Abfrage.

Im unten dargestellten Quellcodeausschnitt der Datei artenfundort.php enthält die URL die Parameter `Active Layer` und `Query`. Die Angabe des aktiven Layers ist notwendig, um dem Server mitzuteilen, auf welchen Layer die Abfrage angewendet werden soll. Damit die Quadranten, in denen die gewählte Art vorkommt, farbig hervorgehoben werden, werden ihre Object_IDs im „Query-Parameter“ an den Kartendienst übergeben.

```
<!--Erzeugung der URL-->
<span style="font-size:1.2em"><br><br><br>Verschaffen Sie sich
 einen Überblick über das Vorkommen der Art:   
</span>

<a href="http://ripsweb.lfu.bwl.de/rips/artenfundort/viewer.htm?
ActiveLayer=7&Query=OBJECT_ID%20in%20(9999,<?
$Sql= "SELECT DISTINCT TK_NAME, ID FROM AE_FUNDORT_V where
 ART_NR = '$art_lfunrn'ORDER BY TK_NAME";
$resultORT = OCI_Parse($db,$Sql);
$query1 = OCI_Execute($resultORT);
   while(oci_fetch_row($resultORT)){
      echo oci_result($resultORT,"ID") . ",";
   }
   ?>9999)" target="_blank"><span style="font-size:1.2em">
Hier geht's zur Karte</span></a><br>
```

Die Skriptsprache PHP wird hier zur dynamischen Erstellung der URL benötigt. Immer gleich ist dabei der erste Teil der URL, bis zur Zeichenfolge <?. Alles was zwischen dieser

Zeichenfolge und der ?>-Zeichenfolge steht, wird vom PHP-Parser interpretiert. Zwischen diesen Zeichen steht die Abfrage der IDs per SQL, die in den PHP-Code eingebettet ist. Diese greift auf die View AE_FUNDORT_V zu, die auf dem Datenbankschema ARTIS angelegt wurde. Das Ergebnis der Abfrage, in diesem Fall die Object_IDs der TK-Quadranten, wird in die URL eingetragen. Die entstehende URL sieht dann beispielsweise so aus:

```
http://ripsweb.lfu.bwl.de/rips/artenfundort/viewer.htm?
ActiveLayer=7&Query= OBJECT_ID%20in%20(9999,102,105,204,9999)
```

Damit immer eine syntaktisch korrekte URL erzeugt wird, muss am Anfang und Ende der Aufzählung in Klammern eine Zahl stehen. In diesem Fall wurde die fiktive Ziffernfolge 9999 verwendet, da diese Zahl sicher nicht als Object_ID vorkommt. Nur so können die Object_IDs hintereinander, durch Komma getrennt, aufgelistet und hinter dem letzten Wert eine schließende Klammer gesetzt werden. Diese sorgt dafür, dass die URL auch korrekt beendet wird.

Abbildung 6.8 zeigt das Ergebnis einer Anfrage an den Kartendienst.

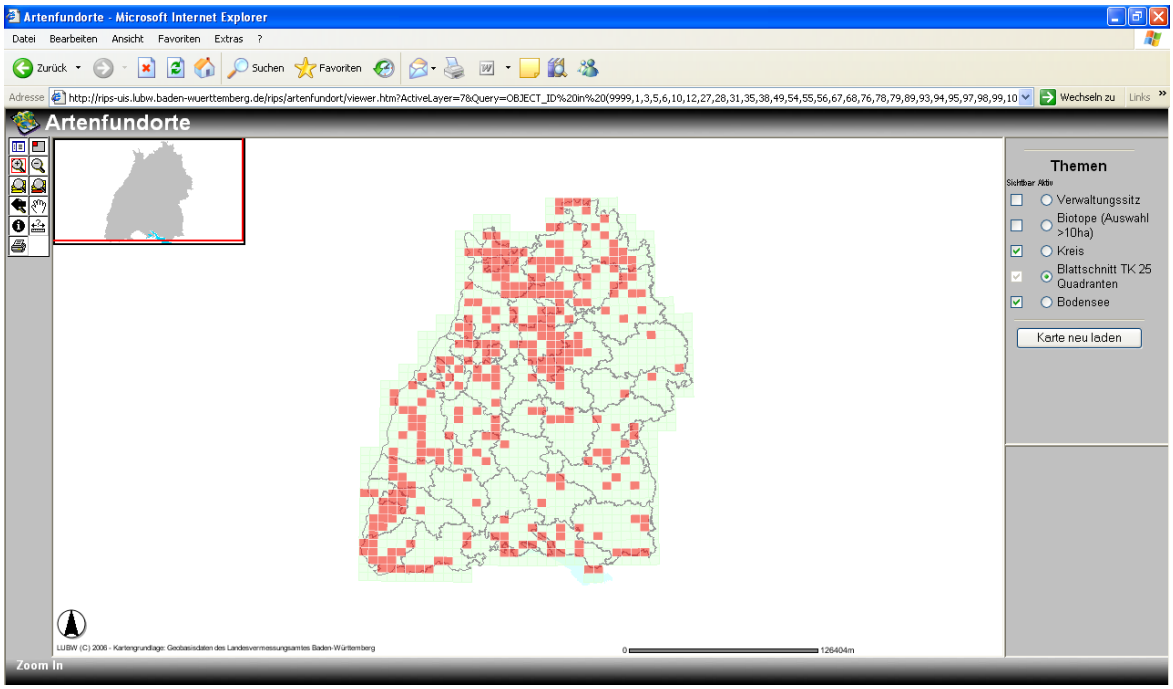


Abbildung 6.8: Ergebnisdarstellung im HTML Viewer

In der endgültigen Version des Kartenviewers sind noch kleinere Änderungen am Aussehen des Viewers durchgeführt worden (vergleiche dazu Abbildung 6.3 und Abbildung 6.8). So wurden die nicht benötigten Werkzeuge aus der Werkzeugleiste entfernt, die Beschriftungen in deutscher Sprache formuliert und die Copyrightangabe berichtigt.

6.6 Fazit der Implementierung des Kartendienstes mit ArcIMS

Grundsätzlich ist die Erstellung eines Kartendienstes mit ArcIMS als komfortable, sichere und einfach modifizierbare Lösung zu bezeichnen. Die Implementierung des Dienstes hat gezeigt, dass die serverseitige Systemumgebung bereits sehr gut an die Erfordernisse dieser Software angepasst ist. Da es sich bei PHP um eine serverseitig ausgeführte Skriptsprache handelt, muss auf dem Server ein so genannter Parser installiert und konfiguriert sein.
Die Erstellung des Dienstes und seine Veröffentlichung im Intranet/Internet sind damit unter den gegebenen Voraussetzungen gut zu realisieren. Auch die Einbettung in das im Intranet bestehende Kartendienstangebot sollte kein Problem darstellen.

Schwierigkeiten gab es leider bei der Umsetzung der farbigen Hervorhebung der Blattschnittquadranten. Prinzipiell bietet der HTML Viewer die Möglichkeit, das Ergebnis einer Abfrage oder Selektion farbig hervorzuheben. Standardmäßig kann die Abfrage im Kartendienst vom Benutzer selbst mithilfe einer Art „Formular“ durchgeführt werden. In Abbildung 6.9 ist dieses dargestellt. Es kann über das rechts über dem Formular abgebildete Werkzeug, das in der Werkzeugleiste der Standardversion zu finden ist, aufgerufen werden.

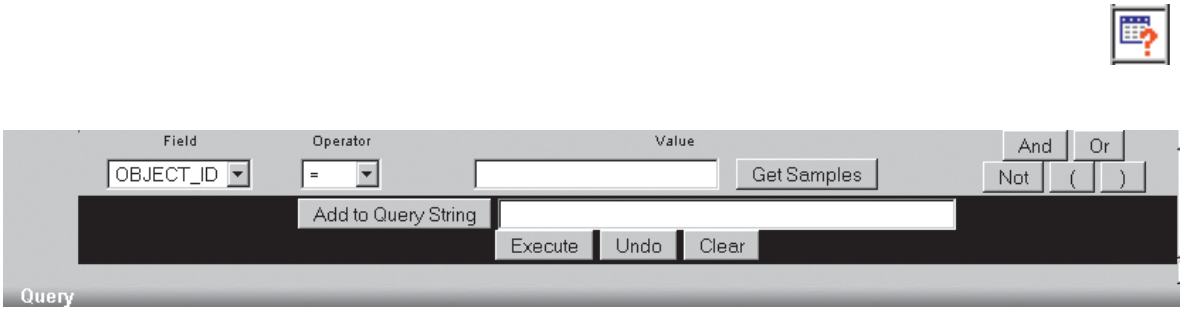


Abbildung 6.9: Query Formular des ArcIMS HTML Viewers (ESRI Standardversion)

Soll das Ergebnis der Selektion nun in der URL übergeben werden und die Anzeige der Quadranten bereits beim Aufruf der Karte erfolgen, muss wie bereits erwähnt, auf die Methode der *Start-Up URL* zurückgegriffen werden. Im Laufe der Entwicklung dieses Projektes hat sich jedoch herausgestellt, dass die Hervorhebung der Quadranten nur bis zu einer gewissen Anzahl an übergebenen Object_IDs möglich ist. Da dieser Sachverhalt auch für zukünftige Projekte von großer Bedeutung sein kann, soll darauf im Folgenden näher eingegangen werden.

Da zunächst vermutet wurde, dass die Hervorhebung aufgrund der zu hohen Anzahl von Object_IDs in der URL nicht funktioniert, wurde zuerst die Clientseite überprüft. Hierzu wurde der Kartendienst in verschiedenen Browsern getestet. Es zeigte sich, dass die Reaktion der Browser auf eine zu lange URL unterschiedlich ist. Vorangestellt werden muss, dass von Seiten des HTTP Protokolls keine Beschränkung der Anzahl der Zeichen in der URL besteht. Um dem Darstellungsproblem weiter auf den Grund zu gehen, wurde zusätzlich im Internet recherchiert.
Unter `http://www.boutell.com/newfaq/misc/urllength.html` findet sich eine Aufstellung über die von den verschiedenen Browsern akzeptierte Anzahl der Zeichen in der URL. Bis auf den Microsoft Internet Explorer, dessen maximal erlaubte URL Länge tatsächlich bei 2083

Zeichen liegt, haben alle Browser eine Begrenzung, die weit über der Anzahl von Zeichen liegt, die in Fall des Kartendienstes in der URL übergeben würden. Im Gegensatz zu der Aussage des Autors der oben genannten Internetseite steht jedoch meine Erfahrung mit der Reaktion des Internet Explorers auf eine zu lange URL: Er schreibt, dass eine Überschreitung der maximalen Zeichenanzahl zu einer klaren Fehlermeldung führt. Im Falle dieses Kartendienstes ist es jedoch so, dass der Browser keine Fehlermeldung ausgibt, sondern die Karte darstellt, jedoch ohne Hervorhebung der Quadranten. Dies liegt daran, dass die URL einfach an der Stelle abgeschnitten wird, an der die maximale Anzahl von Zeichen erreicht ist. Da für eine vollständige URL jedoch die beendende Klammer benötigt wird, kann der Aufruf nicht korrekt durchgeführt werden.

Ganz anders dagegen die Reaktion des Browsers Firefox. Dieser gibt bei zu langer URL die Fehlermeldung: "`Es wurde versucht, auf ein Token zuzugreifen, das nicht vorhanden ist.`" aus. Aufgrund der Aussage bezüglich der erlaubten URL Länge auf der oben genannten Internetseite und dem Test auf weiteren Browsern kann wahrscheinlich ausgeschlossen werden, dass es sich bei der Fehlermeldung um eine Meldung von Seiten des Browsers handelt. Daher muss der Fehler an einer anderen Stelle gesucht werden.

Als weitere Fehlerquellen kommen der *Webserver*, hier der Microsoft Internet Information Services (IIS), das *Tomcat Servlet* sowie der *Application Server* und der *Spatial Server* von ArcIMS in Betracht.
Um den Fehler zu lokalisieren, wurde zunächst das Fehler-Logging des *Webservers* auf höchster Logstufe aktiviert. In der Protokolldatei (Logfile) werden dann alle auf dem *Webserver* eingehenden Anfragen und ausgeführten Arbeitsschritte und natürlich auch Fehler protokolliert. Die Fehlermeldungen werden vom *Webserver* in einem Zahlencode, dem so genannten „Statuscode", verschlüsselt ausgegeben. Dem Statuscode kann dann entnommen werden, ob eine Anforderung erfolgreich ausgeführt wurde oder nicht. Im letzteren Fall kann auch der genaue Grund für eine fehlgeschlagene Anforderung angegeben werden. Tabelle 6.2 zeigt eine Zusammenstellung der möglichen IIS-Statuscodes.

Gruppen von Statuscodes	Beschreibung
1xx - Informational (Informationen)	Vorläufige Antworten. Der Client kann vor der regulären Antwort, mehrere 1xxx - Antworten erhalten
2xx - Success (Erfolgreich)	Anforderung des Clients an den Server wurde erfolgreich akzeptiert
3xx - Redirection (Umleitung)	Weitere Vorgänge müssen ausgeführt werden, um die Anforderung vollständig zu bearbeiten.
4xx - Client Error (Clientfehler)	Ein Fehler auf Clientseite ist aufgetreten.
5xx - Server Error (Serverfehler)	Der Server kann die Anforderung nicht ausführen, weil ein Fehler aufgetreten ist.

Tabelle 6.2: Statuscodes des IIS Webservers

Jede dieser Statuscode-Gruppen enthält zahlreiche Unterkategorien, die den Status genauer spezifizieren. Weitere Informationen hierzu sind im Support Center der Firma Microsoft unter dem Stichwort „IIS Statuscodes“ auf der der Webseite `http://support.microsoft.com/` zu finden.
Die Analyse der Log-Dateien – nach dem Aufruf des Kartendienstes bei einer zu langen URL – zeigte, dass es sich um einen „internen Serverfehler“ handelt (siehe Abbildung 6.10, Fehlercode: 500). Dies bestätigt wiederum die Aussage, dass der Fehler nicht auf Seiten des Clients ausgelöst wird. Des Weiteren ist in der Log-Datei erkennbar, dass der *Webserver* die korrekte URL erhält.

```
2006-12-01 11:27:38 W3SVC1 10.41.253.1 GET /rips/artenfundort/viewer.htm
ActiveLayer=7&Query=OBJECT_ID%20in%20(9999,7,12,15,19,25,30,31,34,36,38,50,
54,56,61,65,67,68,74,76,80,87,89,95,97,98,99,100,103,104,105,106,107,109,110,111,114,
123,124,125,127,128,129,130,132,141,145,147,148,149,150,161,162,163,164,166,170,172,
174,176,181,185,191,192,201,202,205,211,213,215,217,219,229,230,231,235,237,238,239,
240,241,242,243,245,247,248,251,252,253,256,259,260,261,263,266,267,268,278,285,286,
292,293,294,295,297,298,299,300,301,302,303,304,305,306,307,308,309,310,311,312,313,
314,315,317,318,320,330,333,335,336,338,339,340,345,346,348,349,350,351,353,355,356,
357,358,359,360,361,362,363,364,365,366,367,368,369,370,371,372,374,375,376,378,379,
,381,383,391,393,394,395,396,397,398,399,400,401,402,403,404,405,406,407,408,411,417,
418,419,420,421,422,423,424,425,426,427,428,429,430,431,432,433,434,435,436,437,438,
439,440,441,442,443,453,454,455,456,457,458,459,460,461,462,463,464,465,466,467,468,
469,470,471,472,473,476,477,479,480,482,484,485,486,487,488,489,490,491,492,493,494,
495,496,497,498,500,502,503,504,505,506,517,521,522,523,524,525,526,527,528,529,530,
531,532,533,534,535,536,537,538,540,541,543,545,548,549,551,552,555,558,559,560,563,
565,566,567,568,569,570,572,583,585,586,587,588,589,590,591,592,593,594,595,596,597,
598,599,600,601,602,603,604,605,606,607,608,609,612,613,614,616,618,620,624,625,626,
629,630,631,632,633,634,636,637,638,639,645,649,650,651,652,653,654,655,656,657,658,
659,660,661,662,663,664,665,666,667,668,669,670,671,672,673,675,676,678,686,687,689,
692,693,694,696,709,710,711,712,713,714,715,716,717,718,719,720,721,722,723,724,725,
726,727,728,729,730,731,732,733,734,735,736,737,738,739,740,741,742,743,744,745,750,
754,756,757,758,764,773,774,775,776,777,778,779,780,781,782,783,784,785,786,787,788
,789,790,791,792,793,794,795,796,797,798,799,800,801,802,803,804,806,810,820,826,
828,833,835,836,837,838,839,840,841,842,843,844,845,846,847,848,849,850,851,852,853,
854,855,856,857,858,859,860,861,862,863,864,865,866,868,869,871,872,887,892,901,902,
903,904,905,906,907,908,909,910,911,912,913,914,915,916,917,918,919,920,921,922,923,
924,925,926,927,929,930,931,934,936,942,943,949,951,952,953,961,963,965,966,967,968,
969,970,971,972,973,974,975,976,978,979,980,985,986,989,990,995,999,1003,1004,1005,
1007,1008,1012,1013,1014,1015,1017,1020,1023,1025,1026,1027,1028,1029,1030,1031,1032,
1033,1034,1035,1036,1037,1038,1039,1040,1041,1042,1043,1044,1046,1047,1048,1049,1050,
1057,1058,1059,1060,1065,1066,1067,1068,1069,1070,1071,1072,1076,1077,1079,1080,1081,
1082,1083,1084,1085,1089,1090,1091,1092,1093,1094,1095,1096,1097,1098,1099,1100,1101,
1102,1103,1104,1105,1106,1107,1108,1109,1110,1113,1114,1115,1116,1117,1118,1120,1121,
1122,1123,1124,1126,1128,1129,1130,1131,1132,1133,1134,1135,1136,1138,1139,1140,1141,
1142,1143,1144,1145,1146,1147,1148,1153,1156,1157,1158,1159,1160,1161,1162,1163,1164,
1165,1166,1167,1168,1170,1172,1175,1176,1177,1178,1181,1182,1189,1190,1194,1197,1199,
1201,1202,1203,1204,1205,1206,1209,1210,1212,1217,1218,1221,1222,1225,1226,1229,1233,
1234,1237,9999) 80 - 10.40.40.142
Mozilla /5.0+(Windows;+U;+Windows+NT+5.0;+de;+rv:1.8.1)+Gecko/20061010+Firefox/2.0
500 0 1008
```

Abbildung 6.10: Ausschnitt aus der Log-Datei des IIS

Um die Vermutung zu überprüfen, dass es sich um einen Fehler bei der Verarbeitung der Anfrage auf Seiten des *Webservers* bzw. der Serverkomponente *Tomcat* handeln könnte, wurden auch die Log-Dateien des *Application Servers* kontrolliert. In den Log-Dateien dieses Servers, der in der Systemarchitektur hinter dem *Webserver* angesiedelt ist, ist keine eingehende Anfrage zu erkennen. Daraus lässt sich schließen, dass die Anfrage erst gar nicht auf dem *Application Server* ankommt und daher auch nicht an den *Spatial Server* weitergeleitet werden kann (siehe auch Abbildung 5.5). Dementsprechend kann auch keine Karte vom *Spatial Server* erzeugt werden. Dies bestätigt auch die Untersuchung des Output-Verzeichnisses, in dem bei

erfolgreicher Anfragebearbeitung die Karte im PNG-Format abgelegt werden müsste. Somit kann auch der *Spatial Server* als Fehlerverursacher ausgeschlossen werden.
Weitergehende Untersuchungen der Fehlerursache müssten sich daher auf das Tomcat Servlet der Apache Software Foundation konzentrieren.

7 Zusammenfassung und Ausblick

Die vorliegende Diplomarbeit lässt sich inhaltlich in zwei Themenkomplexe gliedern. Während sich der erste Teil der Arbeit mit der Übernahme von Artdaten in das neu entstehende Arteninformationssystem ARTIS beschäftigt, ist der zweite Teil der Visualisierung der Artdaten in einem Kartendienst gewidmet. Beide Teile enthalten Kapitel, die sich zunächst mit der Konzeption der Aufgabe beschäftigen, aber auch Kapitel, welche die Vorgehensweise bei der Realisierung erläutern.

Ziel des ersten Teilbereichs war es, ein Konvertierungswerkzeug für den Import der Daten der § 24a-Biotopkartierung in das Oracle-Datenbankschema ARTIS zu schaffen. Dabei ging es nicht darum, alle Daten der Kartierung, die bislang im Schema ALBIS abgelegt sind, zu übernehmen, sondern nur die Daten für den Import herauszufiltern, die auch Informationen zu Arten enthalten. Die Analyse des Schemas ALBIS, die zu Beginn der Arbeit stand, zeigte, dass Daten zu Artenfunden, Fundorten und Kartierern hauptsächlich in drei Tabellen zu finden sind. Neben der strukturellen und inhaltlichen Analyse des Quellschemas ALBIS war auch die genaue Analyse der Struktur des Zielschemas ARTIS notwendig. Diese Arbeitsschritte, die eher konzeptionellen Charakters sind, nahmen viel Zeit in Anspruch und waren für das Gelingen des Projektes von großer Bedeutung.
Dem Analyseteil folgte die Auseinandersetzung mit der technischen Umsetzungsmöglichkeiten des Datenimports. Für die Entwicklung des Konvertierungswerkzeugs wurden zwei Lösungen näher betrachtet und in der Ausarbeitung beschrieben. Der eigenen Programmierung eines Werkzeugs mit Visual Basic in der Entwicklungsumgebung Visual Studio 6 stand die Verwendung einer bereits bestehenden Softwarekomponente, dem Oracle Warehouse Builder 10.2.0.1, gegenüber. Die Abwägung der Vor- und Nachteile beider Lösungen ergab, dass für das Projekt, so wie es im Rahmen der Diplomarbeit realisiert werden sollte, die Programmierung mit Visual Basic geeigneter war.
Nachdem diese Grundsatzentscheidung getroffen war, galt es die fachlichen Anforderungen, die an das Werkzeug gestellt wurden, programmtechnisch umzusetzen. In Kapitel 4 wird der Aufbau des Konvertierungsprogramms beschrieben. Zur Veranschaulichung wurden in den Text immer wieder Quellcodeausschnitte wichtiger Funktionen eingebunden. Diese haben ebenso wie die Ablaufdiagramme zum Ziel, dem Leser die Funktionen des Importprogramms zu erläutern. Die Ablaufdiagramme sind nach Prozeduren unterteilt dargestellt und aus Übersichtsgründen im Anhang zu finden. Dort befindet sich ebenfalls der gesamte in der Diplomarbeit geschriebene Quellcode.
Im Rahmen der Arbeit wurden mit dem Konvertierungswerkzeug bereits zahlreiche Datensätze in das Schema ARTIS importiert. Da bis ca. 4 Wochen vor Abgabe von der Fachabteilung noch Änderungen an den Daten der § 24a-Biotopkartierung vorgenommen wurden, konnte erst dann mit dem Import begonnen werden. Insgesamt sollen etwa 1,8 Millionen Datensätze in ARTIS übernommen werden. Da die Netzwerk- und Datenbankinfrastruktur nicht zulässt, alle Daten gleichzeitig in einem Importvorgang in das Zielschema zu übernehmen, werden jeweils nur Importvorgänge ausgeführt, die ca. 3000-5000 Datensätze umfassen.

Diese Vorgehensweise benötigt zwar relativ viel Zeit, stellt jedoch den korrekten Import der Daten sicher. Es lässt sich festhalten, dass der Import von 3000 Datensätzen etwa 30 Minuten benötigt. Allerdings steigt die Importdauer überproportional zur Anzahl der zu importierenden Datensätze. Durch die Einträge in der Log-Datei können die Importvorgänge jederzeit kontrolliert und nachvollzogen werden.

In der Entwicklungsumgebung Visual Studio 6 ist ein Assistent vorhanden, der auf einfache Weise die Zusammenstellung eines Pakets ermöglicht, das alle zur Ausführung eines Programms benötigten Dateien enthält. Das Programm kann dann über ein einfaches Setup auf jedem beliebigen PC installiert werden. Die einzige Anforderung an den Computer ist, dass für die Ausführung des Programms ein ODBC-Treiber für Oracle installiert sein muss, über den das Programm eine Verbindung zur Datenbank herstellen kann. Dazu wird auf einem Windows PC unter Systemsteuerung→Verwaltung→Datenquellen (ODBC) ein System-DSN (Data Source Name) mit dem Namen der Datenbank eingerichtet, auf die der Treiber dann zugreift. Als großer Vorteil der Programmierung gegenüber der Verwendung des Oracle Warehouse Builders ist die Tatsache anzusehen, dass das Programm leicht zu installieren ist und aufgrund der benutzerfreundlichen Anwendungsoberfläche ohne Vorkenntnisse verwendet werden kann.

Ziel des zweiten Teils der Diplomarbeit war es, einen Kartendienst aufzubauen, der die Visualisierung der Artenfunde ermöglicht. Auch hier musste zunächst ermittelt werden, welche Anforderungen von fachlicher Seite an den Dienst gestellt werden. Unter Beachtung dieser Anforderungen wurde dann eine Auswahl verschiedener Map-Server-/ Map-Client-Systeme untersucht. Die umfangreiche Beschreibung der einzelnen Systeme, denen sich Kapitel 5.3 widmet, war notwendig, um das Zusammenspiel von Map Server/Map Client und der weiteren Komponenten möglichst allgemein verständlich zu erläutern. Aufgrund der Komplexität der einzelnen Systeme wurde jedoch nicht immer in der gleichen Tiefe auf die Komponenten und Funktionalitäten eingegangen. Bereits während der Einarbeitung in die einzelnen Lösungen stellte sich heraus, dass der Artenfund-Kartendienst am besten mit der Software ArcIMS zu realisieren sein wird. Daher wurde diese Lösung auch etwas ausführlicher beschrieben als die anderen Systeme. HTML Viewer und Image Server von ArcIMS bieten alle für diesen Kartendienst benötigten Funktionalitäten.
Der Einstieg in den Kartendienst erfolgt über eine Artenliste auf einer Webseite. Die Daten dieser Liste werden mittels der Skriptsprache PHP, die über die OCI-Schnittstelle auf die Oracle Schemata zugreift, immer aktuell aus der Datenbank geladen. Es werden immer nur die Tier- und Pflanzenarten zur Auswahl angeboten, die zum Zeitpunkt der Seitenaufrufs im Schema ARTIS vorhanden sind. Die Darstellung der Artenfundorte einer vom Benutzer ausgewählten Art wird im Kartendienst durch die Hervorhebung der TK25-Quadranten, in denen die Fundorte liegen, realisiert. Diese „Verschleierung“ des genauen Fundorts ist notwendig, da die Datenquelle auch besonders gefährdete Tier- und Pflanzenarten enthält, die nicht punktgenau dargestellt werden dürfen.
Durch die Übergabe der Objekt IDs dieser Quadranten in der URL wird bereits beim Aufruf des Kartendienstes das Ergebnis in der Karte angezeigt. Kurz vor Fertigstellung der Diplomarbeit stellte sich heraus, dass aus bislang ungeklärter Ursache nur eine gewisse Anzahl von Argumenten in der URL übergeben werden kann. Die zur Lösung des Problems unternommenen Anstrengungen sind in Kapitel 6.6 ausführlich dargestellt.
Da die Anzeige der Artenfunde jedoch für einen großen Teil der in ARTIS gespeicherten Art-

daten funktioniert, wurde bis zuletzt intensiv nach der Lösung des Fehlers gesucht. Da dieser höchstwahrscheinlich nicht von einer der ArcIMS-Komponenten verursacht wird, und der auf Basis von ArcIMS erstellte Kartendienst zum jetzigen Zeitpunkt am besten in die gesamte Infrastruktur und das bestehende Angebot der LUBW passt, wäre die weitere Fehlersuche auf Seiten des Webservers wünschenswert.
Es ist jedoch möglich, dass zu einem späteren Zeitpunkt weitere Anforderungen an den Kartendienst gestellt werden, die sich auf Basis von PHP, HTML und ArcIMS nicht mehr umsetzen lassen. Dann müsste in jedem Fall nach einer Alternative gesucht werden.
Zum jetzigen Zeitpunkt ist der Kartendienst nur dem Fachpublikum zugänglich. Nach der Plausibilisierung der bereits importierten Daten ist jedoch geplant, den Kartendienst auch der Öffentlichkeit anzubieten. Außerdem sollen möglichst bald weitere Datenquellen in ARTIS importiert werden. Enthält das Datenbankschema ARTIS dann mehrere Datenquellen, wäre es nützlich, wenn der Karte entnommen werden könnte aus welcher Datenquelle die dargestellten Fundorte stammen. So könnte für jede Datenquelle dann eine eigene Signatur verwendet werden, die dann in der Karte die Lage des Fundort angibt. Dies lässt sich allerdings nur bei einer punktgenauen Darstellung der Fundorte in der Karte realisieren.
Der Kartendienst ist, so wie er im Rahmen der Diplomarbeit realisiert wurde, sicherlich nur als Prototyp zu sehen. Vielfältige Erweiterungsmöglichkeiten sind denkbar, die jedoch in jedem Fall zunächst mit der Fachabteilung abgestimmt werden müssen und deren Nutzen überprüft werden muss. Vorstellbar wäre zum Beispiel, dass zukünftig eine Unterscheidung in der Darstellung der Rote Liste Arten und der „Nicht Rote Liste Arten“ gemacht wird. Interessant wäre außerdem, wenn die Kartierer die Möglichkeit hätten, Artvorkommen direkt über das Internet in eine Karte einzutragen. Die Koordinaten der markierten Fundorte könnten dann direkt in das Datenbankschema ARTIS übernommen werden. Dies erfordert aber in jedem Fall die Einbindung erweiterter GIS-Funktionalitäten.

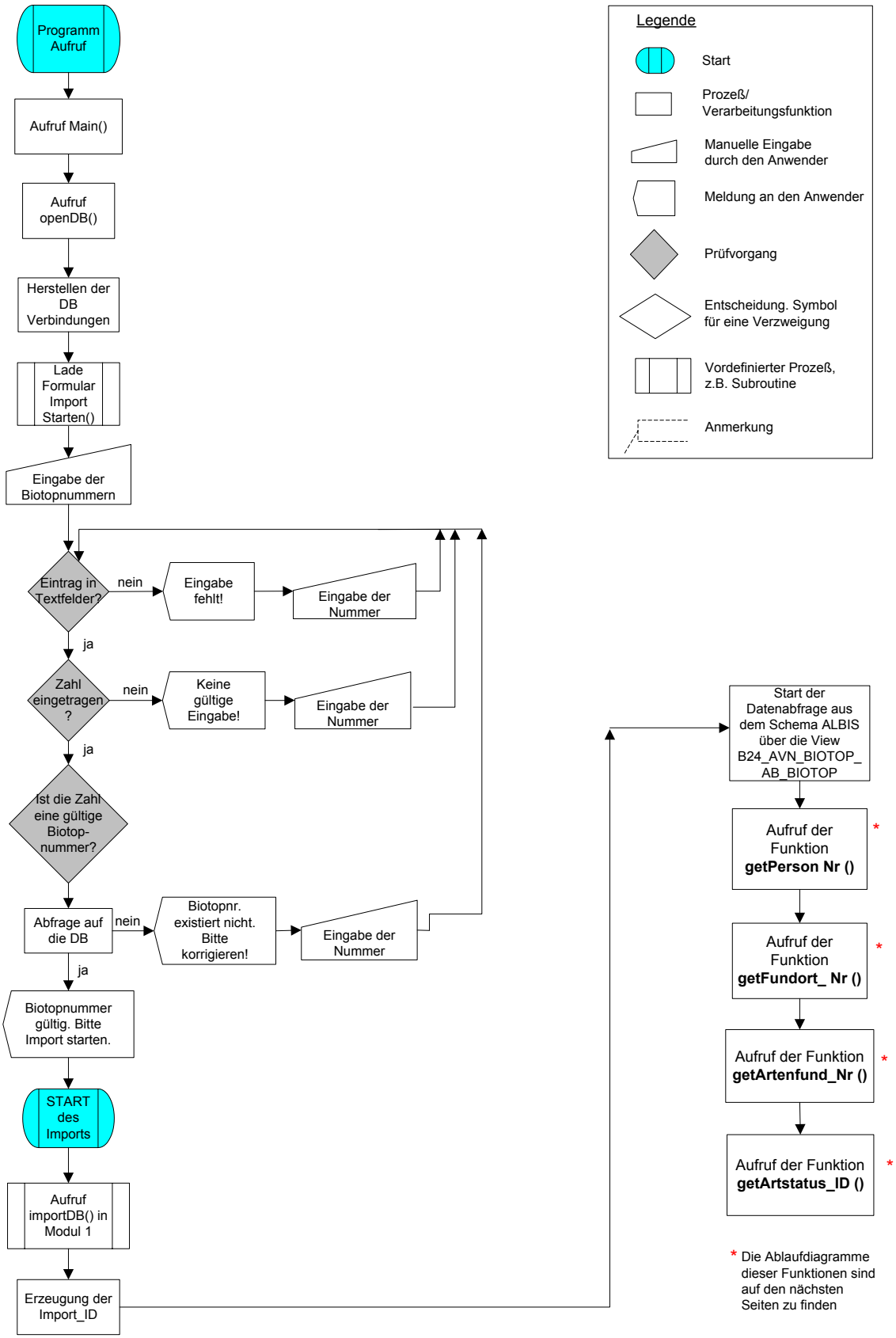
Schlussendlich lässt sich festhalten, dass es sich beim Thema Artenschutz um ein sehr umfangreiches und sensibles Thema handelt, das jedoch mehr denn je unsere Aufmerksamkeit verdient hat. Durch den Import der Daten der § 24a-Biotopkartierung in das Datenbankschema ARTIS im Rahmen dieser Diplomarbeit wurde ein weiterer wichtiger Schritt in Richtung des Aufbaus eines zentralen Arteninformationssystems gegangen. Die Erstellung eines Kartendienstes zur Visualisierung der Artenfunde bietet zukünftig auch interessierten Bürgerinnen und Bürgern die Möglichkeit, die landesweit gesammelten Daten aus dem Bereich Artenschutz einzusehen. Damit erfüllt die LUBW nicht zuletzt die Vorgaben des neuen Umweltinformationsgesetzes.

Anhang A

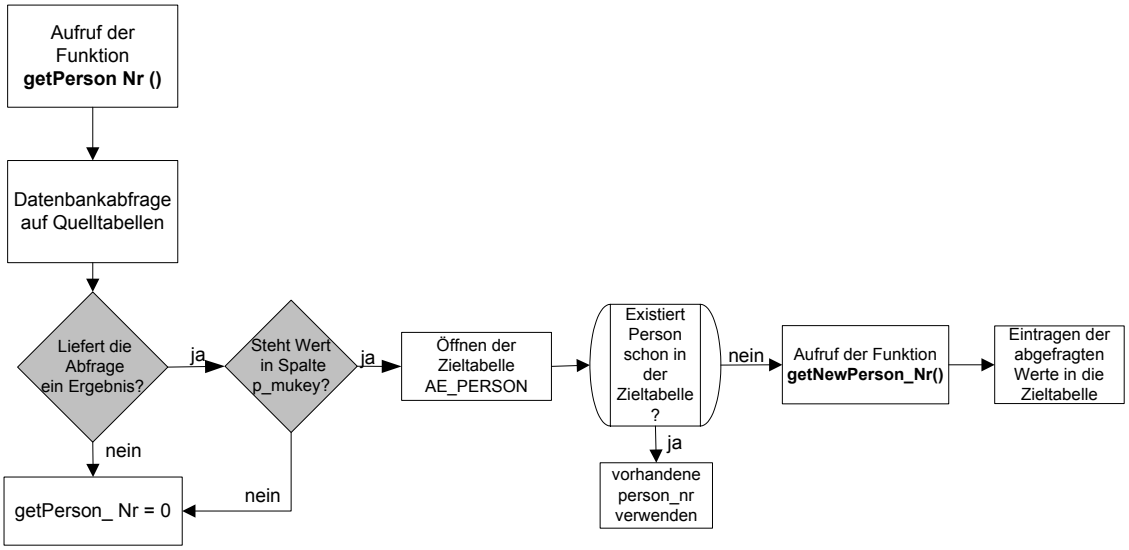
Ablaufdiagramme zum Visual Basic Programm

Aufgrund seiner Größe befindet sich das erste Diagramm auf der nächsten Seite.

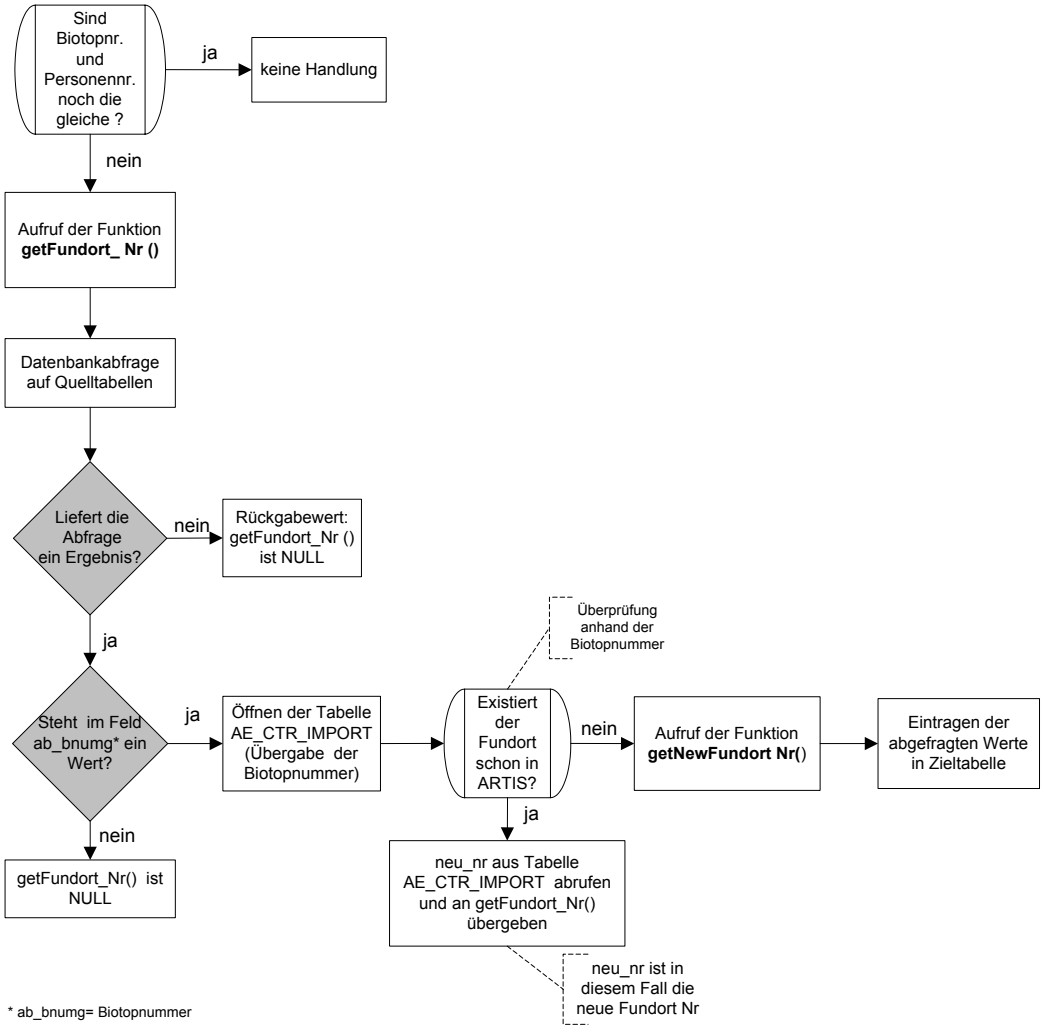
Übersicht über den Programmablauf



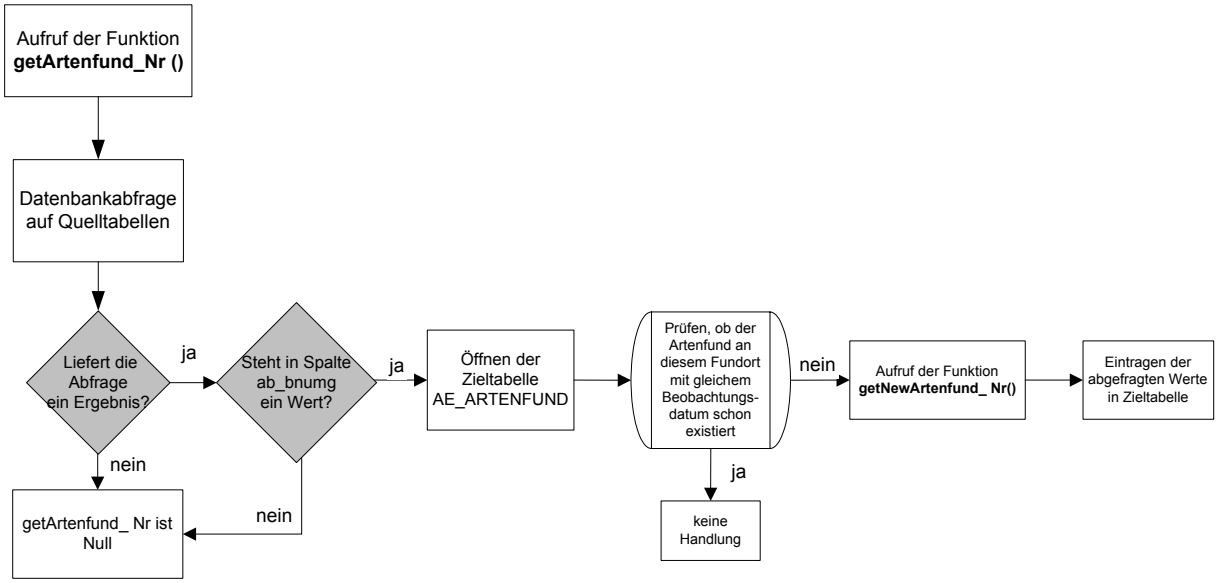
Ablauf der Funktionen getPerson_Nr() und getNewPerson_Nr()



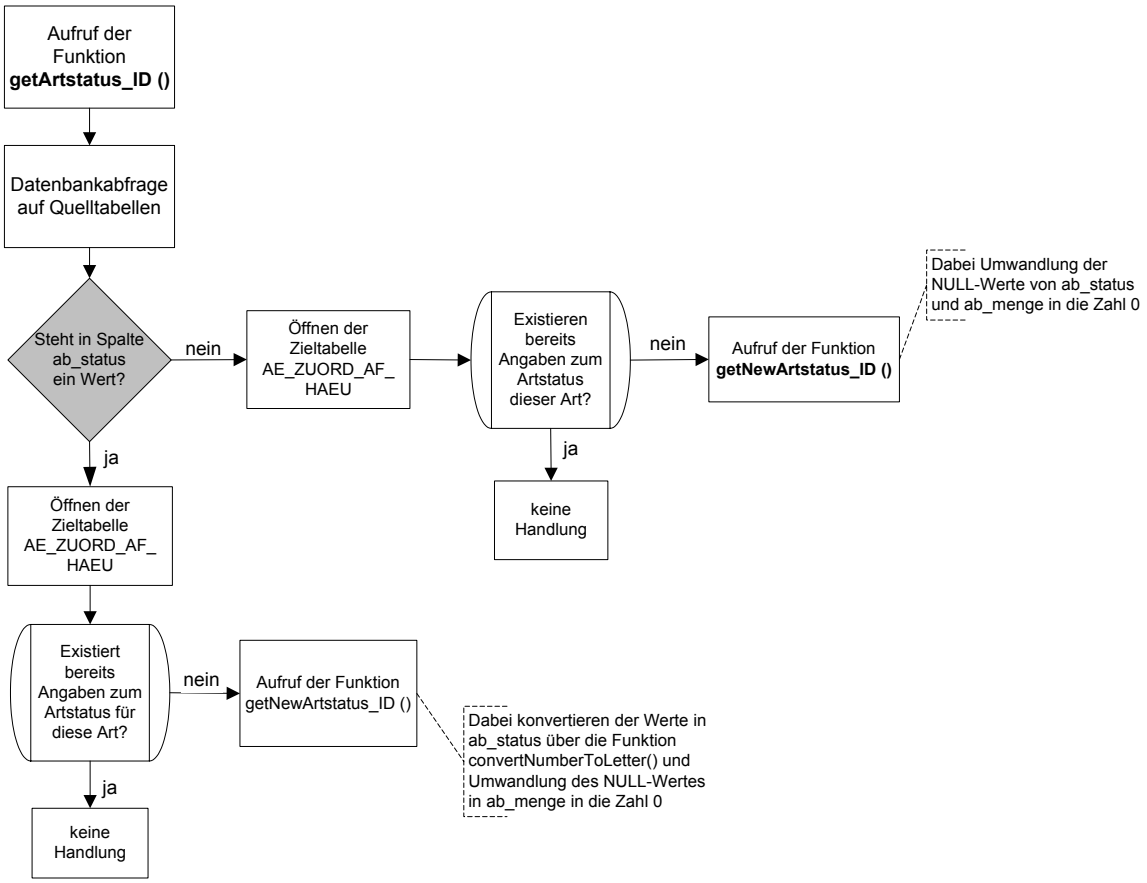
Ablauf der Funktionen getFundort_Nr() und getNewFundort_Nr()



Ablauf der Funktionen getArtenfund_Nr() und getNewArtenfund_Nr()



Ablauf der Funktionen getArtstatus_ID() und getNewArtstatus_ID()



Anhang B

Quellcode

B.1 Quellcode des Importprogramms in Visual Basic

```
#QUELLCODE DES MODUL 1

'ALLGEMEINER DEKLARATIONSTEIL

Attribute VB_Name = "Module1"
Option Explicit     'Variablen müssen explizit definiert werden
Public dbQuelle As New ADODB.Connection
Public dbZiel As New ADODB.Connection
Public countNewPerson, countDatensaetze, countNewArtenfund, countNewFundort,
       countNewArtstatus_ID, anzahlDatensaetze As Double
Public getNewImport_ID
Public bolErrorOccured As Boolean
Public Abbrechen As Boolean
Const c_sModuleFileName As String =
 "D:\Diplomarbeit\Importsoftware VB\Code\Module1.bas"

******************************************************
'HAUPTPROZEDUR

Public Sub Main()
  On Error GoTo ErrorHandler

   Call openDB
   Load Import_starten
   Import_starten.Show

  Exit Sub
ErrorHandler:
  HandleError True, "Main " & c_sModuleFileName & " " &
   GetErrorLineNumberString(Erl), Err.Number, Err.Source,
   Err.Description, 4
End Sub
******************************************************
'ÖFFNEN DER DATENBANKVERBINDUNGEN

Public Sub openDB() 'Verbindungsparameter angeben
  On Error GoTo ErrorHandler
```

```
dbQuelle.Provider = "OraOLEDB.Oracle.1"
dbQuelle.Properties("Data Source") = "UIST2_DB"
dbQuelle.Properties("Persist Security Info") = "False"
dbQuelle.Properties("User ID") = "albis"
dbQuelle.Properties("Password") = "***"
dbQuelle.Open
dbQuelle.CursorLocation = adUseClient

dbZiel.Provider = "OraOLEDB.Oracle.1"
dbZiel.Properties("Data Source") = "UIST2_DB"
dbZiel.Properties("Persist Security Info") = "False"
dbZiel.Properties("User ID") = "artis"
dbZiel.Properties("Password") = "xxx"
dbZiel.Open
dbZiel.CursorLocation = adUseClient


  Exit Sub
ErrorHandler:
  HandleError True, "openDB " & c_sModuleFileName & " " &
   GetErrorLineNumberString(Erl),
   Err.Number, Err.Source, Err.Description, 4
End Sub
*********************************************************
'SCHLIEßEN DER DATENBANKVERBINDUNGEN

Public Sub closeDB()
  On Error GoTo ErrorHandler

    dbQuelle.Close
    dbZiel.Close

  Exit Sub
ErrorHandler:
  HandleError True, "closeDB " & c_sModuleFileName & " " &
   GetErrorLineNumberString(Erl),Err.Number, Err.Source, Err.Description, 4
End Sub
*************************************************************************
'IMPORT PROZEDUR

Public Sub importDB(von As String, bis As String)
  On Error GoTo ErrorHandler

    Dim rsQuelle As New ADODB.Recordset
    Dim Person_Nr As String
    Dim fundort_nr  As String
    Dim Artenfund_Nr As String
    Dim oldBIOTOP_NR As String
    Dim oldPerson_NR As String
    Dim oldAB_ALFUNRN As String
    Dim oldArtenfund_NR As String
    Dim oldART_BEOB_DAT As String
    Dim art_beob_dat As String
```

```
    Dim artstatus_id As String
    Dim ctr_Import As String
    Dim rsID As New ADODB.Recordset


    'Neue Import ID erzeugen
    rsID.Open "SELECT (MAX(IMPORT_ID)+1)AS newImportID FROM AE_CTR_IMPORT", dbZiel
    getNewImport_ID = rsID("newImportID")
    rsID.Close

     rsQuelle.Open "SELECT * FROM B24AVN_BIOTOP_AB_BIOTOP WHERE AB_BNUMG
     between '" + von + "' and  '" + bis + "' order by ab_bnumg,
     qpmukey desc nulls last", dbQuelle  'lesender Zugriff auf die Quelltabelle"

    anzahlDatensaetze = rsQuelle.RecordCount

    Do Until rsQuelle.EOF

    DoEvents
    If Abbrechen = True Then
    ImportLog ("Import wurde vom User abgebrochen.")
    ImportLog ("**********************************")
    Exit Sub
    End If

'ANZAHL DER IMPORTIERTEN DATENSAETZE WIRD GEZÄHLT
countDatensaetze = countDatensaetze + 1
Import_starten.lblFortschritt.Caption = "Datensatz " & countDatensaetze &
 " von " & anzahlDatensaetze & " wird gerade importiert."
Import_starten.Refresh


Person_Nr = getPerson_Nr(rsQuelle("AB_BNUMG"), rsQuelle("AB_ALFUNRN"),
            rsQuelle("ART_BEOB_DAT"))

'Abfrage, ob Biotopnummer und Person_Nr noch die gleiche sind wie im
vorherigen Durchlauf
If ((rsQuelle("AB_BNUMG") = oldBIOTOP_NR) And (Person_Nr = oldPerson_NR)) Then
 'Do nothing
Else
 'neue Fundort_Nr erzeugen
  fundort_nr = getFundort_Nr(rsQuelle("AB_BNUMG"), Person_Nr)
End If

  Artenfund_Nr = getArtenfund_Nr(rsQuelle("AB_BNUMG"), fundort_nr, Person_Nr,
                 rsQuelle("AB_ALFUNRN"), CStr(rsQuelle("Art_Beob_Dat")))

  artstatus_id = getArtstatus_ID(rsQuelle("AB_BNUMG"), Artenfund_Nr,
                 rsQuelle("AB_ALFUNRN"))

  oldBIOTOP_NR = rsQuelle("AB_BNUMG")
  oldPerson_NR = Person_Nr
  oldAB_ALFUNRN = rsQuelle("AB_ALFUNRN")
```

```
  oldART_BEOB_DAT = CStr(rsQuelle("Art_Beob_Dat"))
  oldArtenfund_NR = Artenfund_Nr

  rsQuelle.MoveNext

Loop

ImportLog ("Es wurde(n) " + CStr(countNewPerson) + " Person(en) neu angelegt.")
ImportLog ("Es wurde(n) " + CStr(countNewArtenfund) +
           " Artenfund(e) neu angelegt.")
ImportLog ("Es wurde(n) " + CStr(countNewFundort) +
           " Fundort(e) neu angelegt.")
ImportLog ("Es wurde(n) " + CStr(countNewArtstatus_ID) +
           " Artstatus/Artstati neu angelegt.")

If countNewPerson = 0 And countNewArtenfund = 0 And countNewFundort = 0 And
 countNewArtstatus_ID = 0 Then
  ImportLog ("Import mit Import_ID " + CStr(getNewImport_ID) + " wurde nicht
             ausgeführt, da Daten bereits in ARTIS vorhanden waren.")
  ImportLog ("************************************************************")
Else

  ImportLog ("Import mit Import Nr " + CStr(getNewImport_ID) +
              " wurde durchgeführt.")
  ImportLog ("************************************************************")

End If
rsQuelle.Close

  Exit Sub
ErrorHandler:
 HandleError False, "importDB " & c_sModuleFileName & " " &
 GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description, 4
End Sub
**************************************************************************
'ABRUF DER PERSONENDATEN (KARTIERER)

Private Function getPerson_Nr(biotop_nr As String, art_nr As String,
 art_beob_dat As String) As String
  On Error GoTo ErrorHandler


  Dim rsQuelle As New ADODB.Recordset
  Dim rsZiel As New ADODB.Recordset

  rsQuelle.Open "SELECT A.AB_BNUMG,a.AB_JAHR, B.*, C.WOHNORT,
   D.ANREDE from B24A_AB_BIOTOP A , B24AVN_PERSON B, B24AVN_WOHNORTENTSCHL C,
    B24AVN_ANREDE_SL D where A.AB_QPMUKEY = B.P_MUKEY and
    C.KENNZIFFER (+) = B.P_KENNZ and A.AB_BNUMG='" + biotop_nr + "' and
    D.ANREDE_SL = B.P_ANREDE(+) and ab_alfunrn = '" + art_nr + "' and
    ab_jahr = '" + art_beob_dat + "'", dbQuelle

    If rsQuelle.RecordCount = 0 Then
```

```
        getPerson_Nr = 0    'Rückgabewert
    ImportLog ("Keine Person in Quelltabelle angegeben.")

    Else
        If IsNull(rsQuelle("P_MUKEY")) Then
            getPerson_Nr = 0
        Else
            'Person in der QuellDB gefunden
            ' --> existiert die Person schon in der Ziel-Tabelle?
            rsZiel.Open "SELECT * FROM AE_PERSON WHERE NACHNAME " &
             convertNullToSqlNull(rsQuelle("P_NNAME")) &
             " AND VORNAME " & convertNullToSqlNull(rsQuelle("P_VNAME")) &
             " AND KENNUNG=0", dbZiel

            ' existiert diese noch nicht:
            If rsZiel.RecordCount = 0 Then
                getPerson_Nr = getNewPerson_Nr(rsQuelle("P_NNAME"),
                 rsQuelle("P_VNAME"), rsQuelle("ANREDE"), rsQuelle("P_TITEL"),
                 rsQuelle("P_STRHNR"), rsQuelle("P_POSTF"),
                 rsQuelle("P_TELEFON"), rsQuelle("P_FAX"), rsQuelle("WOHNORT"),
                 convertNullToNumber(rsQuelle("P_UKEY")))
            Else
                'wenn Person schon existiert -> vorhandene Person_Nr verwenden
                'Eintrag in ImportLog Datei
                ImportLog ("Person mit der PERSON_NR "
                 + CStr(rsZiel("PERSON_NR")) + " existiert schon.
                 Vorhandene PERSON_NR wird verwendet!")
            getPerson_Nr = rsZiel("PERSON_NR")
            End If
        End If
    End If

  Exit Function
ErrorHandler:
  HandleError False, "getPerson_Nr " & c_sModuleFileName & " " &
   GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description, 4
    End Function
'*************************************************************************
'ABRUF DER INFORMATIONEN ZUM ARTENFUNDORT
Private Function getFundort_Nr(l_biotop_nr As String, l_Person_Nr As String)
  As String
  On Error GoTo ErrorHandler

    Dim rsQuelle As New ADODB.Recordset
    Dim rsZiel As New ADODB.Recordset

    rsQuelle.Open "SELECT BIO_NAME, AB_BNUMG, TK_NR, TK_QUADRANT, BIO_R,
     BIO_H, GEOLOGIE_ID, BIO_ERFDAT, BIO_AENDDAT FROM B24AVN_BIOTOP_AB_BIOTOP A
     where AB_BNUMG = '" + l_biotop_nr + "'", dbQuelle

     If rsQuelle.RecordCount = 0 Then
        getFundort_Nr = "NULL"    'Rückgabewert
        ImportLog ("Kein Fundort in Quelltabelle angegeben.")
```

```
     Else
        If IsNull(rsQuelle("AB_BNUMG")) Then
            getFundort_Nr = "NULL"
        Else
        rsZiel.Open "select * from ae_ctr_import where alt_nr =
          '" + l_biotop_nr + "'", dbZiel

        ' existiert dieser noch nicht:
            If rsZiel.RecordCount = 0 Then
              getFundort_Nr = getNewFundort_Nr(rsQuelle("BIO_NAME"),
              rsQuelle("TK_Nr"), rsQuelle("TK_QUADRANT"), rsQuelle("BIO_R"),
              rsQuelle("BIO_H"), rsQuelle("GEOLOGIE_ID"),
              rsQuelle("BIO_ERFDAT"), rsQuelle("BIO_AENDDAT"),
              l_biotop_nr, l_Person_Nr)
            Else
              'wenn Fundort schon existiert -> vorhandene Fundort_Nr verwenden
               getFundort_Nr = rsZiel("NEU_NR")
              'Eintrag in ImportLog Datei
               ImportLog ("Fundort mit der Fundortnummer "
                + CStr(rsZiel("NEU_NR")) +
                " (Biotopnummer " + CStr(rsZiel("ALT_NR")) + ")  existiert schon.
                Vorhandene FUNDORT_NR = " + CStr(rsZiel("NEU_NR")) +
                "  wird verwendet!")
            End If
         End If
     End If
    rsQuelle.Close

  Exit Function
ErrorHandler:
  HandleError False, "getFundort_Nr " & c_sModuleFileName & " " &
  GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description, 4
End Function
*************************************************************************
'ABRUF DER INFORMATIONEN ZUM ARTENFUND

Private Function getArtenfund_Nr(l_biotop_nr As String, l_fundort_nr As String,
 l_Person_Nr As String, l_LFUNRN As String, art_beob_dat As String) As String
  On Error GoTo ErrorHandler

    Dim rsQuelle As New ADODB.Recordset
    Dim rsZiel As New ADODB.Recordset

    rsQuelle.Open "SELECT AB_BNUMG, AB_ALFUNRN,ART_BEOB_DAT, BIO_ERFDAT,
    BEMERKUNG FROM B24AVN_BIOTOP_AB_BIOTOP WHERE AB_BNUMG='" + l_biotop_nr + "'
    AND AB_ALFUNRN='" + l_LFUNRN + "'", dbQuelle

    If rsQuelle.RecordCount = 0 Then
        getArtenfund_Nr = "NULL"    'Rückgabewert
     Else
        If IsNull(rsQuelle("AB_BNUMG")) Then
            getArtenfund_Nr = "NULL"
        Else
```

```
        ' exisitiert Artenfund an diesem Fundort mit gleichem Beobachtungsdatum
          schon in Zieltabelle?

        rsZiel.Open "SELECT * FROM AE_ARTENFUND WHERE FUNDORT_NR =
        '" + l_fundort_nr + "' and LFUNRN = '" + l_LFUNRN + "' and
        Beobachtungsdatum = '" + art_beob_dat + "'", dbZiel

          'existiert dieser noch nicht:
           If rsZiel.RecordCount = 0 Then
             getArtenfund_Nr = getNewArtenfund_Nr(l_LFUNRN, rsQuelle("Bemerkung"),
             rsQuelle("BIO_ERFDAT"), art_beob_dat, l_fundort_nr, l_Person_Nr)
           Else
          'existiert diese schon:
              ImportLog ("Artenfund mit der LFUNRN = " + l_LFUNRN + " und dem
              Beobachtungsdatum = " + art_beob_dat + " existiert am Fundort mit
              der Nr = " + l_fundort_nr + " schon.")
             getArtenfund_Nr = rsZiel("Artenfund_Nr")

          End If
       End If
    End If
    rsQuelle.Close

  Exit Function
ErrorHandler:
  HandleError False, "getArtenfund_Nr " & c_sModuleFileName & " " &
  GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description, 4
End Function
*************************************************************************
'ABRUF DER INFORMATIONEN STATUS DER ART

Private Function getArtstatus_ID(l_biotop_nr, l_Artenfund_Nr As String,
 l_LFUNRN As String) As String
  On Error GoTo ErrorHandler

 Dim rsQuelle As New ADODB.Recordset
 Dim rsZiel As New ADODB.Recordset

 rsQuelle.Open "select a.AB_STATUS, a.AB_MENGE from b24a_ab_biotop a where
  a.AB_BNUMG='" + l_biotop_nr + "' and a.AB_ALFUNRN='" + l_LFUNRN + "'", dbQuelle

   If IsNull(rsQuelle("AB_STATUS")) Then
   rsZiel.Open "SELECT * FROM AE_ZUORD_AF_HAEU WHERE ARTENFUND_NR =
   '" + l_Artenfund_Nr + "'", dbZiel
        If rsZiel.RecordCount = 0 Then
           getArtstatus_ID = getNewArtstatus_ID(convertNullToNumber
           (rsQuelle("AB_STATUS")), convertNullToNumber(rsQuelle("AB_MENGE")),
           l_Artenfund_Nr)
        Else
            'do nothing
       End If
   Else
   rsZiel.Open "SELECT * FROM AE_ZUORD_AF_HAEU WHERE ARTENFUND_NR =
```

```
   '" + l_Artenfund_Nr + "'", dbZiel
      If rsZiel.RecordCount = 0 Then
            getArtstatus_ID = getNewArtstatus_ID(convertNumberToLetter
            (rsQuelle("AB_STATUS")), convertNullToNumber(rsQuelle("AB_MENGE")),
            l_Artenfund_Nr)
      Else
          'do nothing
      End If
   End If

rsQuelle.Close

Exit Function
ErrorHandler:
  HandleError False, "getArtstatus_ID " & c_sModuleFileName & " " &
  GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description, 4
End Function
*************************************************************************
'EINTRAG IN ARTIS-TABELLE AE_PERSON

Private Function getNewPerson_Nr(nachname, vorname, anrede, titel, strasse,
 postfach, telefon, fax, wohnort, p_ukey) As String
  On Error GoTo ErrorHandler


    Dim rsZiel As New ADODB.Recordset
    Dim rsID As New ADODB.Recordset

    rsID.Open "SELECT (MAX(PERSON_NR)+1) as newID FROM AE_PERSON", dbZiel
        getNewPerson_Nr = rsID("newID")
    rsID.Close

    rsZiel.Open "SELECT * FROM AE_PERSON WHERE PERSON_NR=0", dbZiel,
     adOpenDynamic, adLockOptimistic 'schreibender Zugriff auf die Zieltabelle

    rsZiel.AddNew

    ImportLog ("Neue Person mit PERSON_NR= " + CStr(getNewPerson_Nr) +
               " wurde angelegt.")
    countNewPerson = countNewPerson + 1

    rsZiel("PERSON_NR") = getNewPerson_Nr
    rsZiel("NACHNAME") = nachname
    rsZiel("VORNAME") = vorname
    rsZiel("ANREDE") = anrede
    rsZiel("KENNUNG") = 0
    rsZiel("TITEL") = titel
    rsZiel("STRASSE") = strasse
    rsZiel("POSTFACH") = postfach
    rsZiel("TELEFON") = telefon
    rsZiel("FAX") = fax
    rsZiel("ORT") = wohnort
```

```
    rsZiel.Update
    rsZiel.Close

    'Eintrag in ae_ctr_import
    Call write_ctr_Import("1", CStr(p_ukey), getNewPerson_Nr)


  Exit Function
ErrorHandler:
  HandleError False, "getNewPerson_Nr " & c_sModuleFileName & " " &
  GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description, 4
End Function
*************************************************************************
' EINTRAG IN ARTIS-TABELLE AE_FUNDORT

Private Function getNewFundort_Nr(Fundort_Name, TK25, TK_Quadrant, Rechtswert,
 Hochwert, Geologie_ID, Bio_Erfdat, Bio_Aenddat, l_biotop_nr As String,
 l_Person_Nr As String) As String
  On Error GoTo ErrorHandler


 Dim rsZiel As New ADODB.Recordset
 Dim rsFundID As New ADODB.Recordset

 rsFundID.Open "SELECT (MAX(FUNDORT_NR)+1) as newFUND_ID FROM AE_FUNDORT", dbZiel
    getNewFundort_Nr = rsFundID("newFUND_ID")
 rsFundID.Close

    rsZiel.Open "SELECT * FROM AE_FUNDORT WHERE FUNDORT_NR=0", dbZiel,
     adOpenDynamic, adLockOptimistic

    rsZiel.AddNew

    ImportLog ("Neuer Fundort mit FUNDORT_NR= " + getNewFundort_Nr + "
    (Biotopnummer: " + l_biotop_nr + ") wurde angelegt.")
    countNewFundort = countNewFundort + 1

    rsZiel("FUNDORT_NR") = getNewFundort_Nr
    rsZiel("FUNDORT_NAME") = Fundort_Name
    rsZiel("TK25") = TK25
    rsZiel("TK_QUADRANT") = TK_Quadrant
    rsZiel("RECHTSWERT") = Rechtswert
    rsZiel("HOCHWERT") = Hochwert
    rsZiel("GEOLOGIE_ID") = Geologie_ID
    rsZiel("AENDERUNGSDATUM") = Bio_Aenddat
    rsZiel("ANLAGEDATUM") = Bio_Erfdat
    If l_Person_Nr <> 0 Then rsZiel("PERSON_NR") = l_Person_Nr

    rsZiel.Update
    rsZiel.Close

    Call write_ctr_Import("3", l_biotop_nr, getNewFundort_Nr)
```

```
  Exit Function
ErrorHandler:
  HandleError False, "getNewFundort_Nr " & c_sModuleFileName & " " &
  GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description, 4
End Function
*************************************************************************
'EINTRAG IN ARTIS-TABELLE AE_ARTENFUND

Private Function getNewArtenfund_Nr(LFUNRN As String, Bemerkung, Bio_Erfdat,
 art_beob_dat As String, l_fundort_nr As String, l_Person_Nr As String) As String
  On Error GoTo ErrorHandler


  Dim rsZiel As New ADODB.Recordset
  Dim rsArtfundID As New ADODB.Recordset

  rsArtfundID.Open "SELECT (MAX(ARTENFUND_NR)+1) as newAF_ID FROM AE_ARTENFUND",
                   dbZiel
  getNewArtenfund_Nr = rsArtfundID("newAF_ID")
  rsArtfundID.Close

  rsZiel.Open "SELECT * FROM AE_ARTENFUND WHERE ARTENFUND_NR = 0", dbZiel,
              adOpenDynamic, adLockOptimistic

  rsZiel.AddNew

  ImportLog (" Neuer Artenfund mit ARTENFUND_NR = " + getNewArtenfund_Nr + "
             und LFUNRN = " + LFUNRN + " und Beobachtungsdatum = " + art_beob_dat +
             " wurde angelegt.")
  countNewArtenfund = countNewArtenfund + 1

  rsZiel("ARTENFUND_NR") = getNewArtenfund_Nr
  rsZiel("FUNDORT_NR") = l_fundort_nr
  rsZiel("LFUNRN") = LFUNRN
  rsZiel("BEMERKUNG") = Bemerkung
  rsZiel("ANLAGEDATUM") = Bio_Erfdat
  rsZiel("BEOBACHTUNGSDATUM") = art_beob_dat
  rsZiel("POTENZIELL") = 1
  If l_Person_Nr <> 0 Then rsZiel("PERSON_NR") = l_Person_Nr

  rsZiel.Update
  rsZiel.Close

  Call write_ctr_Import("4", LFUNRN, getNewArtenfund_Nr)


  Exit Function
ErrorHandler:
  HandleError False, "getNewArtenfund_Nr " & c_sModuleFileName & " " &
  GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description, 4
End Function
```

```
**************************************************************************
'EINTRAG IN ARTIS-TABELLE AE_ZUORD_AF_HAEU

Private Function getNewArtstatus_ID(AB_STATUS, AB_MENGE, l_Artenfund_Nr)
 As String
  On Error GoTo ErrorHandler

Dim rsZiel As New ADODB.Recordset

rsZiel.Open "SELECT * FROM AE_ZUORD_AF_HAEU WHERE ARTSTATUS_ID = '10'", dbZiel,
            adOpenDynamic, adLockOptimistic
rsZiel.AddNew

ImportLog ("Neue Status_ID = " + convertNullToNumber(AB_STATUS) + " von
           ARTENFUND_NR = " + l_Artenfund_Nr + " wurde angelegt.")
countNewArtstatus_ID = countNewArtstatus_ID + 1

rsZiel("ARTENFUND_NR") = l_Artenfund_Nr
If IsNull(AB_STATUS) Then rsZiel("ARTSTATUS_ID") = 0 Else
  rsZiel("ARTSTATUS_ID") = AB_STATUS
rsZiel("HAEUFIGKEITSTYP_ID") = "ga"
rsZiel("HAEUFIGKEIT_ID") = AB_MENGE
rsZiel("HERBA") = 0
rsZiel("KARTEI") = 0
rsZiel("LITERATUR") = 0

rsZiel.Update
rsZiel.Close

  Exit Function
ErrorHandler:
  HandleError False, "getNewArtstatus_ID " & c_sModuleFileName & " " &
   GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description, 4
End Function
**************************************************************************
'EINTRÄGE IN DIE LOG DATEI

Public Sub ImportLog(ByVal strError As String)   'Log-Datei erzeugen
  On Error GoTo ErrorHandler


  Dim text As String, fso, f, txf As Object

  Set fso = CreateObject("Scripting.FileSystemObject") 'FILE SYSTEM OBJECT setzen

    If fso.FileExists(App.Path & "\import.log") Then
        Set f = fso.GetFile(App.Path & "\import.log")
        Set txf = f.OpenAsTextStream(1, -2)
        text = txf.ReadAll
        txf.Close
        text = Right(text, 10000000)
    End If
```

```
    text = text & Now() & ": " & "Datensatz Nr." + CStr(countDatensaetze) + ": "
     + strError 'Now() gibt aktuelles Datum an

    Set txf = fso.OpenTextFile(App.Path & "\import.log", 2, True, 0)
    txf.WriteLine CStr(text)
    txf.Close

  Exit Sub
ErrorHandler:
  HandleError True, "ImportLog " & c_sModuleFileName & " " &
  GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description, 4
End Sub
*************************************************************************
'DATENSATZWEISER EINTRAG IN IMPORTTABELLE

Private Function write_ctr_Import(tab_id As String, alt_nr As String,
  neu_nr As String) As String
  On Error GoTo ErrorHandler


  Dim rsZiel As New ADODB.Recordset

  rsZiel.Open "SELECT * FROM AE_CTR_IMPORT WHERE TAB_ID = 0", dbZiel,
   adOpenDynamic, adLockOptimistic
  rsZiel.AddNew

  rsZiel("IMPORT_ID") = getNewImport_ID
  rsZiel("TAB_ID") = tab_id
  rsZiel("ALT_NR") = alt_nr
  rsZiel("NEU_NR") = neu_nr
  rsZiel("HERKUNFT_NR") = 3
  rsZiel("DATUM") = Date  'Datumsangabe ohne Uhrzeit
  rsZiel("VORGANG") = "I"

  rsZiel.Update
  rsZiel.Close


  Exit Function
ErrorHandler:
  HandleError False, "write_ctr_Import " & c_sModuleFileName & " " &
  GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description, 4
End Function
*************************************************************************
'KONVERTIEREN VON NULL IN 0

Private Function convertNullToNumber(X) As String
  On Error GoTo ErrorHandler

  If IsNull(X) Then
   convertNullToNumber = "0"
Else
   convertNullToNumber = CStr(X)
```

```
  Exit Function
  ErrorHandler:
  HandleError False, "convertNullToNumber " & c_sModuleFileName & " " &
  GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description, 4
End Function
*************************************************************************

Private Function convertNullToSqlNull(X) As String
    If IsNull(X) Then
      convertNullToSqlNull = "IS NULL"
    Else
     convertNullToSqlNull = "='" + CStr(X) + "'"
End Function
*************************************************************************
' UMWANDLUNG DES ARTSTATUS SCHLÜSSELS

Private Function convertNumberToLetter(X) As String
  On Error GoTo ErrorHandler

    If X = 0 Then convertNumberToLetter = "0"
    If X = 1 Then convertNumberToLetter = "IN"
    If X = 2 Then convertNumberToLetter = "SY"
    If X = 3 Then convertNumberToLetter = "3"
    If X = 4 Then convertNumberToLetter = "AN"
    If X = 5 Then convertNumberToLetter = "BR"
    If X = 6 Then convertNumberToLetter = "BV"
    If X = 7 Then convertNumberToLetter = "DU"
    If X = 8 Then convertNumberToLetter = "IR"
    If X = 9 Then convertNumberToLetter = "WI"
    If X = Null Then convertNumberToLetter = "0"

  Exit Function

ErrorHandler:
  HandleError False, "convertNumbertoLetter " & c_sModuleFileName & " " &
  GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description, 4
End Function
```

```
#QUELLCODE DES FORMULARMODULS "Import_starten"

Option Explicit
Public knopf As Boolean
*************************************************************************
Private Sub cmdBeenden_Click()
    Unload Me
End Sub
*************************************************************************
Private Sub Import_starten_Click()
Import_starten.Enabled = False ' Import Button erst aktiv, wenn Prüfvorgang
                               erfolgreich war

If Len(txtVon) = 0 Then MsgBox ("Eingabe fehlt!"): Exit Sub
If Not IsNumeric(txtVon) Then MsgBox ("Keine gültige Eingabe!"): Exit Sub
If Not IsNumeric(txtBis) Then MsgBox ("Keine gültige Eingabe!"): Exit Sub
Call Module1.importDB(txtVon, txtBis)

End Sub
*************************************************************************
Private Sub Löschen_Click()
txtVon.text = ""
txtBis.text = ""
End Sub
*************************************************************************
Private Sub Prüfen_Click()
Dim rsQuelle As New ADODB.Recordset
Dim txtBis_neu
Dim txtVon_neu
Dim rsQuelle2 As New ADODB.Recordset

If Len(txtVon) = 0 Then MsgBox ("Eingabe fehlt!"): Exit Sub
If Not IsNumeric(txtVon) Then MsgBox ("Keine gültige Eingabe!"): Exit Sub
If Not IsNumeric(txtBis) Then MsgBox ("Keine gültige Eingabe!"): Exit Sub

rsQuelle.Open "Select * from B24AVN_BIOTOP_AB_BIOTOP where AB_BNUMG =
              '" + txtVon.text + "'", dbQuelle
If rsQuelle.RecordCount = 0 Then
  Call MsgBox("Linke Biotopnumer existiert nicht! Bitte korrigieren Sie die
              Nummer.", vbOKOnly): Exit Sub
End If
rsQuelle.Close

rsQuelle2.Open "Select * from B24AVN_BIOTOP_AB_BIOTOP where AB_BNUMG =
               '" + txtBis.text + "'", dbQuelle
If rsQuelle2.RecordCount = 0 Then
   Call MsgBox("Rechte Biotopnumer existiert nicht! Bitte korrigieren Sie die
               Nummer.", vbOKOnly): Exit Sub
End If
rsQuelle2.Close
Import_starten.Enabled = True

If txtVon > txtBis Then
```

```
    txtBis_neu = txtVon
    txtVon_neu = txtBis
    txtBis = txtBis_neu
    txtVon = txtVon_neu
End If

    rsQuelle.Open "Select * from B24AVN_BIOTOP_AB_BIOTOP where AB_BNUMG =
                  '" + txtVon.text + "'", dbQuelle
    If rsQuelle.RecordCount = 0 Then
        Call MsgBox("Linke Biotopnumer existiert nicht! Bitte korrigieren Sie
                     die Nummer.", vbOKOnly): Exit Sub
    End If

    rsQuelle2.Open "Select * from B24AVN_BIOTOP_AB_BIOTOP where AB_BNUMG =
                   '" + txtBis.text + "'", dbQuelle
    If rsQuelle2.RecordCount = 0 Then
        Call MsgBox("Rechte Biotopnumer existiert nicht! Bitte korrigieren Sie
                     die Nummer.", vbOKOnly): Exit Sub
    End If

    Call MsgBox("Biotopnummern sind gültig. Bitte starten Sie den
                Importvorgang.", vbOKOnly): Exit Sub

End Sub
*******************************************************************************
Public Sub cancel_click()
Abbrechen = True
MsgBox ("Import abgebrochen!")

Unload Me
End Sub
```

B.2 Quellcode der Datei artensuche.php

```
<!-- Erzeugung der Auswahllisten der Tier- bzw. Pflanzengruppen und
     wissenschaftlichen Artnamen -->

<?
$suche="1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25";
ini_set ("display_errors","0");

if (!empty($_GET)) {extract($_GET);}
else if (!empty($HTTP_GET_VARS)) {extract($HTTP_GET_VARS);}

if (!empty($_POST)) {extract($_POST);}
else if (!empty($HTTP_POST_VARS)) {extract($HTTP_POST_VARS);} // end if
$db = OCILogon("artis","xxx","uist2_db");
?>

<HTML>
<BODY>
<body bgcolor="#FFFFCC">
<font size=+2>Artensuche über wissenschaftliche Artnamen</font><br><br>
<TABLE cellspacing="0">

<!--ERSTE AUSWAHLLISTE-->
<TR>
<form name ="form_suche" method="post" action="artensuche.php">
<input type="hidden" name="suche" value="<?echo $suche;?>">
<TD>Gruppen von Tier- und Pflanzenarten:</TD>
<TD>

<?
$error=OCI_Error();
if($error) print_r($error);
$suche_field = split(",",$suche);
$sql = "SELECT distinct ARTTYP_DTNAME, ARTTYP from Artis.B24AVN_ARTEN WHERE";
   foreach ($suche_field as $value) {
 $sql .= " ARTTYP = " . $value . " OR";
 }
$sql = substr($sql,0,-3) . " ORDER BY ARTTYP";
?>

<? if (!$artSuche) {
$artSuche = 1;
}?>

<select name="artSuche" size="1" style="font-size: 10pt"
 onChange="javascript:document.getElementsByName('form_suche')[0].submit()">
<?
 $resultTYP = OCI_Parse($db,$sql);
 $query1 = OCI_Execute($resultTYP)
  while(oci_fetch_array($resultTYP)){?>
<option value="<?echo oci_Result ($resultTYP, "ARTTYP");?>"
   <?if ($artSuche == oci_Result($resultTYP,"ARTTYP")) {echo " selected"; }?>>
```

```
       <?echo oci_result($resultTYP, "ARTTYP_DTNAME");?></option>
      <?
      }?>
</select>
</TD>
</form>
</TR>

<!--ZWEITE AUSWAHLLISTE;ANZEIGE ABHÄNGIG VON AUSWAHL IN ERSTER LISTE-->
<TR>
<form name ="objektwahl" method="post" action="artenfundort.php" target="bild">
<TD class="tocs" style="vertical-align:top">Wissenschaftlicher Artname:</TD>
<TD style="vertical-align:top">
<?
if ($artSuche) {
    $sql = "SELECT distinct a.ART_WISSNAME, a.ART_LFUNRN FROM ALBIS.AL_ARTSTAMM_V a,
            B24AVN_ARTEN b WHERE b.LFUNRN = a.ART_LFUNRN AND
            ART_TYP= $artSuche order by ART_WISSNAME";
    $resultART = oci_Parse($db,$sql);
    $result1 = oci_Execute($resultART);
?>
<input type="hidden" name="art_lfunrn"
  value="<?echo oci_Result($resultART,"art_lfunrn");?>">
<input type="hidden" name="sql" value="<?echo $sql;?>">

<select name="art_lfunrn" size="10" style="font-size: 10pt"><?
$result1 = oci_Execute($resultART);
while(oci_fetch_row($resultART)){?>
<option value="<?echo oci_Result($resultART, "ART_LFUNRN");?>">
 <?echo oci_Result($resultART, "ART_WISSNAME");?></option>
<?}
}?>
</select> <br/><br/><input type="submit" value="OK">
</TD>
</form>
</TR>
</TABLE>
<p>
<a href="artensuche_deutsch.php">Zur Suche über deutsche Artnamen</a> <br>

</BODY>
</HTML>
```

B.3 Quellcode der Datei artensuche_deutsch.php

```
<!-- Erzeugung der Auswahllisten der Tier- bzw. Pflanzengruppen und
     deutschen Artnamen -->
<?
$suche="1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25";
ini_set ("display_errors","0");

if (!empty($_GET)) {extract($_GET);}
else if (!empty($HTTP_GET_VARS)) {extract($HTTP_GET_VARS);}

if (!empty($_POST)) {extract($_POST);}
else if (!empty($HTTP_POST_VARS)) {extract($HTTP_POST_VARS);} // end if
$db = OCILogon("artis","xxx","uist2_db");
?>

<HTML>
<BODY>
<body bgcolor="#FFFFCC">
<font size=+2>Artensuche über deutsche Artnamen</font><br><br>
<TABLE cellspacing="0">

<!--ERSTE AUSWAHLLISTE-->
<TR>
<form name ="form_suche" method="post" action="artensuche_deutsch.php">
<input type="hidden" name="suche" value="<?echo $suche;?>">
<TD>Gruppen von Tier- und Pflanzenarten:</TD>
<TD>

<?
$error=OCI_Error();
if($error) print_r($error);

$suche_field = split(",",$suche);
$sql = "SELECT DISTINCT ARTTYP_DTNAME, ARTTYP FROM ARTIS.B24AVN_ARTEN WHERE";
  foreach ($suche_field as $value) {
  $sql .= " ARTTYP = " . $value . " OR";
        }
$sql = substr($sql,0,-3) . " ORDER BY ARTTYP";
?>

<? if (!$artSuche) {
$artSuche = 1;
}?>

<select name="artSuche" size="1" style="font-size: 10pt"
onChange="javascript:document.getElementsByName('form_suche')[0].submit()">
<?
$resultTYP = OCI_Parse($db,$sql);
$query1 = OCI_Execute($resultTYP);
        while(oci_fetch_array($resultTYP)){?>
<option value="<?echo oci_Result ($resultTYP, "ARTTYP");?>"
 <? if ($artSuche == oci_Result($resultTYP,"ARTTYP")) {echo " selected"; }?>>
```

```
 <?echo oci_result($resultTYP, "ARTTYP_DTNAME");?></option>
<?
}?>

</select>
</TD>
</form>
</TR>

<!--ZWEITE AUSWAHLLISTE;ANZEIGE ABHÄNGIG VON AUSWAHL IN ERSTER LISTE-->

<TR>
<form name ="objektwahl" method="post" action="artenfundort.php" target="bild">
<TD class="tocs" style="vertical-align:top">Deutscher Artname:</TD>
<TD style="vertical-align:top">
<?
if ($artSuche) {
$sql = "SELECT DISTINCT a.ART_LFUNRN, b.DTNAME FROM ALBIS.AL_ARTSTAMM_V a, B24AVN_ARTEN b
       WHERE a.ART_LFUNRN = b.LFUNRN and ART_TYP= $artSuche ORDER BY DTNAME";
$resultART = oci_Parse($db,$sql);
$result1 = oci_Execute($resultART);
?>
<input type="hidden" name="lfunrn" value="<?echo oci_Result($resultART,"lfunrn");?>">
<input type="hidden" name="sql" value="<?echo $sql;?>">

<select name="art_lfunrn" size="10" style="font-size: 10pt"><?
$result1 = oci_Execute($resultART);
   while(oci_fetch_row($resultART)){?>
<option value="<?echo oci_Result($resultART, "ART_LFUNRN");?>">
        <?echo oci_Result($resultART, "DTNAME");?></option>
   <?}
   }?>
</select> <br/><br/><input type="submit" value="OK">
</TD>
</form>
</TR>
</TABLE>
<p>
<a href="artensuche.php">Zur Suche über wissenschaftliche Artnamen</a> <br>

</BODY>
</HTML>
```

B.4 Quellcode der Datei artenfundort.php

```
<!-- Angabe von Artnummer, wissenschaftlichem und deutschem Artnamen
     und der Gefährdungskategorie -->
<!-- Auflistung der TK Quadranten, in denen die ausgewählte Art vorkommt -->
<!-- Links für die Weiterleitung zum Kartenviewer und dem Bildarchiv -->

<?

ini_set ("display_errors","0");

if (!empty($_GET)) {extract($_GET);}

else if (!empty($HTTP_GET_VARS)) {extract($HTTP_GET_VARS);}

if (!empty($_POST)) {extract($_POST);}

else if (!empty($HTTP_POST_VARS)) {extract($HTTP_POST_VARS);} // end if
$db = OCILogon("artis","xxx","uist2_db");
?>

<html>
<head>

<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1"/>
<title>Artenfundorte</title>

<!--<link rel="StyleSheet" type="text/css"
href="http://www.um.baden-wuerttemberg.de/Styles.css"/>-->

<style type="text/css">
BODY {
background-color: #FFFFCC;
font-size: 12px;
}

.Header
{
margin: 20px 10px 20px 0px;
padding: 5px 5px 5px 5px;
       font-weight: bold;
background-color: #99CC99;

}

</style>
<!--Javascript File zur Festlegegung der Fenstergröße des Viewers-->
<script type='text/javascript' language='JavaScript'>
function PopUp800x600(targetURL)
{
NewWindow =
window.open(targetURL,'karte',
'width=800,height=600,locationbar=yes,menubar=no,scrollbars=yes,status=yes,
```

```
resizable=no,screenX=10,screenY=10,top=10, left=10');
NewWindow.focus();
}
</script>

</head>
    <body>

<!-- ANGABE DER ART IM GRÜNEN BALKEN -->

<?
$error=OCI_Error();
if($error) print_r($error);

$aSql =  "SELECT LFUNRN, WISSNAME, DTNAME, GEFBADWUE FROM B24AVN_ARTEN
         WHERE LFUNRN='$art_lfunrn'";
$resultART = OCI_Parse ($db, $aSql);
$query = OCI_Execute($resultART);
OCI_Fetch($resultART);
?>
<div class="header"><?echo OCI_Result ($resultART, "LFUNRN")?><br>
                            <?echo OCI_Result ($resultART, "WISSNAME")?><br>
                            <?echo OCI_Result ($resultART, "DTNAME")?><br>
      Gefährdungskategorie: <?echo OCI_Result ($resultART, "GEFBADWUE")?><br>
      </div>


<!-- AUFLISTUNG DER TK-BLÄTTER UND QUADRANTEN -->

<span style="font-size:1.2em">Auflistung der TK25-Quadranten,
                              in denen die gewählte Art vorkommt:<br><br>
<?
$error=OCI_Error();
if($error) print_r($error);

$Sql= "SELECT distinct TK_NAME,ID FROM AE_FUNDORT_V where ART_NR =
      '$art_lfunrn'ORDER BY TK_NAME";
$resultORT = OCI_Parse($db,$Sql);
$query1 = OCI_Execute($resultORT);
while(oci_fetch_array($resultORT)){
    echo oci_result($resultORT,"TK_NAME")?><br><?
}?>

</span>

<span style="font-size:1.2em"><br><br>Verschaffen Sie sich einen Überblick über
                                    das Vorkommen der Art:   </span>
<!--Erzeugung der URL-->
<a href="http://rips-uis.lubw.baden-wuerttemberg.de/rips/artenfundort/viewer.htm?
ActiveLayer=6&Query=OBJECT_ID%20in%20(9999,<?
          $Sql= "SELECT DISTINCT TK_NAME,ID FROM AE_FUNDORT_V where ART_NR =
                '$art_lfunrn'ORDER BY TK_NAME";
          $resultORT = OCI_Parse($db,$Sql);
```

```
          $query1 = OCI_Execute($resultORT);
          while(oci_fetch_row($resultORT)){
              echo oci_result($resultORT,"ID") . ",";
          }
                 ?>9999)" target="_blank"><span style="font-size:1.2em">
                                  Hier geht's zur Karte</span></a><br>
<br>

<!-- BILD ZUR GEWÄHLTEN ART ANZEIGEN -->

<span style="font-size:1.2em"><br>Wollen Sie wissen, wie die Art "
 <?echo OCI_Result ($resultART, "DTNAME")?>" aussieht?</span>  

<a href="http://rips-uis.lubw.baden-wuerttemberg.de/rips/artenfundort/docview.php?
        object_id=<? echo $art_lfunrn;?>&oac=168&ffc=10"target="_blank"
        <span style="font-size:1.2em">Bild anzeigen</span></a><br>

<br>
<a href="artensuche_deutsch.php"><span style="font-size:1.2em">
         Zurück zur Suche über deutsche Artnamen</span></a><br>
<a href="artensuche.php"><span style="font-size:1.2em">
         Zurück zur Suche über wissenschaftliche Artnamen</span></a><br>

<?
ocilogoff($db);
?>


<div class="header"><a class="top" href="#top">nach oben</a></div>

</body>
</html>
```

B.5 Quellcode der Datei artenfundort.axl

```
<!-- Zugriff des Artenfundort-Services auf die Datei artenfundort.axl -->
<!-- Grundlage zur Erstellung der Karte -->
<!-- Definition der Layer, des Layouts der Layer und Angabe der Shape-Dateien -->

<?xml version="1.0" encoding="UTF-8"?>

<ARCXML version="1.1">
  <CONFIG>

<ENVIRONMENT>
  <LOCALE country="DE" language="de" variant="" />
  <UIFONT color="0,0,0" name="SansSerif" size="12" style="regular" />
  <SCREEN dpi="96" />
  </ENVIRONMENT>
<MAP>
  <PROPERTIES>
    <ENVELOPE minx="3373732,9699999997" miny="5254209,109229076"
       maxx="3676781,525" maxy="5532892,065881057" name="Initial_Extent" />
    <MAPUNITS units="meters" />
    <FEATURECOORDSYS id="31467" />
    <FILTERCOORDSYS id="31467" />
    <BACKGROUND color="255,255,255" transcolor="255,255,255"/>

  </PROPERTIES>

  <WORKSPACES>
    <SHAPEWORKSPACE directory="\\itzgs2\fix_data\bw_uebersicht" name="shp_ws-100" />
    <SHAPEWORKSPACE directory="\\itzgs2\fix_data\artenfundort" name="shp_ws-101" />
    <SHAPEWORKSPACE directory="\\itzgs2\fix_data\biotope" name="shp_ws-102" />
    <SHAPEWORKSPACE directory="\\itzgs2\fix_data\biotope" name="shp_ws-103" />
    <SDEWORKSPACE name="sde_ws-0" server="ripssde" instance="port:5151"
      database="" user="lfu_webview" encrypted="true" password="GEOPGWDQUCUNWX" />
  </WORKSPACES>


<!-- RASTERKARTEN -->

<LAYER type="image" name="TK 25" id="1" visible="true" minscale="1:5000"
   maxscale="1:12500">
  <DATASET workspace="sde_ws-0" name="ADMIN.UIS_0100000005800001.RASTER"/>
  <IMAGEPROPERTIES transparency="1" />
  <COORDSYS id="31467" />
</LAYER>

<LAYER type="image" name="TK 50" id="2" visible="true" minscale="1:12500"
   maxscale="1:25000">
  <DATASET workspace="sde_ws-0" name="ADMIN.UIS_0100000005900001.RASTER"/>
  <IMAGEPROPERTIES transparency="1" />
  <COORDSYS id="31467" />
</LAYER>
```

```
<LAYER type="image" name="TK 100" id="3" visible="true" minscale="1:25000"
   maxscale="1:50000">
  <DATASET workspace="sde_ws-0" name="ADMIN.UIS_0100000006000001.RASTER"/>
  <IMAGEPROPERTIES transparency="1" />
  <COORDSYS id="31467" />
</LAYER>

<LAYER type="image" name="T&#220;K 200" id="4" visible="true" minscale="1:50000"
   maxscale="1:125000">
  <DATASET workspace="sde_ws-0" name="ADMIN.UIS_0100000006100001.RASTER"/>
  <IMAGEPROPERTIES transparency="1" />
  <COORDSYS id="31467" />
</LAYER>

<LAYER type="image" name="&#220;K 500" id="5" visible="true" minscale="1:125000"
  maxscale="1:300000">
  <DATASET workspace="sde_ws-0" name="ADMIN.UIS_0100000006200001.RASTER"/>
  <IMAGEPROPERTIES transparency="1" />
  <COORDSYS id="31467" />
</LAYER>

<!-- BODENSEE -->

<LAYER type="featureclass" name="Bodensee" visible="true" id="see"
   minscale="1:400000" maxscale="1:5000000">
  <DATASET name="bodensee_dlm25" type="polygon" workspace="shp_ws-100" />
  <SIMPLERENDERER>
    <SIMPLEPOLYGONSYMBOL  filltype="solid" filltransparency="1.0"
    fillcolor="219,255,255" boundary="false" boundarytype="solid"
    boundarywidth="1" boundarycolor="146,180,242" />
  </SIMPLERENDERER>
<COORDSYS id="31467" />
      </LAYER>


<!-- BLATTSCHNITTE QUADRANTEN-->

<LAYER type="featureclass" name="Blattschnitt TK 25 Quadranten"
   visible="true" id="tk25bs_q">
  <DATASET name="tk25bs_q2" type="polygon" workspace="shp_ws-101" />
  <SIMPLERENDERER>
    <SIMPLEPOLYGONSYMBOL filltype="solid" filltransparency="0,5"
     fillcolor="221,255,217" boundarywidth="1" boundarycolor="221,255,217" />
  </SIMPLERENDERER>
<COORDSYS id="31467" />
      </LAYER>


<!-- VERWALTUNGSGRENZEN-->

<LAYER type="featureclass" name="Kreis" visible="true" id="kreis">
  <DATASET name="krei25" type="polygon" workspace="shp_ws-100" />
  <SIMPLERENDERER>
```

```
    <SIMPLEPOLYGONSYMBOL  filltransparency="0,0" boundarywidth="1"
     boundarycolor="110,110,110" />
  </SIMPLERENDERER>
  <COORDSYS id="31467" />
</LAYER>

<!-- Biotope -->

<LAYER type="featureclass" name="Biotope (Auswahl >10ha)"
  visible="false" id="oac71_10ha" minscale="1:400000" maxscale="1:5000000">
  <DATASET name="oac71_gen_10ha" type="polygon" workspace="shp_ws-103" />
  <SIMPLERENDERER>
    <SIMPLEPOLYGONSYMBOL filltype="solid" filltransparency="1.0"
    fillcolor="153,0,0" boundary="true" boundarycaptype="round" />
  </SIMPLERENDERER>
<COORDSYS id="31467" />
</LAYER>

<LAYER type="featureclass" name="Biotope (Auswahl >5ha)"
  visible="false" id="oac71_5ha" minscale="1:150000" maxscale="1:400000">
  <DATASET name="oac71_gen_5ha" type="polygon" workspace="shp_ws-103" />
  <SIMPLERENDERER>
    <SIMPLEPOLYGONSYMBOL filltype="solid" filltransparency="0,7"
    fillcolor="153,0,0" boundary="true" boundarycaptype="round" />
  </SIMPLERENDERER>
<COORDSYS id="31467" />
</LAYER>

<LAYER type="featureclass" name="Biotope (Auswahl >2ha)"
visible="false" id="oac71_2ha" minscale="1:50000" maxscale="1:150000">
  <DATASET name="oac71_gen_2ha" type="polygon" workspace="shp_ws-103" />
  <SIMPLERENDERER>
    <SIMPLEPOLYGONSYMBOL filltype="solid" filltransparency="0,7"
    fillcolor="153,0,0" boundary="true" boundarycaptype="round" />
  </SIMPLERENDERER>
   <COORDSYS id="31467" />
</LAYER>

<LAYER type="featureclass" name="Biotope (Auswahl >1ha)"
  visible="false" id="oac71_1ha" minscale="1:30000" maxscale="1:50000">
  <DATASET name="oac71_gen_1ha" type="polygon" workspace="shp_ws-103" />
  <SIMPLERENDERER>
     <SIMPLEPOLYGONSYMBOL boundarytransparency="1,0" filltype="solid"
     filltransparency="0,7" fillcolor="153,0,0" boundarycaptype="round" />
  </SIMPLERENDERER>
     <COORDSYS id="31467" />
</LAYER>

<LAYER type="featureclass" name="Biotope (Gesamt)"
  visible="false" id="16" minscale="1:1" maxscale="1:30000">
  <DATASET name="oac71" type="polygon" workspace="shp_ws-103" />
  <SIMPLERENDERER>
    <SIMPLEPOLYGONSYMBOL boundarytransparency="1,0" filltype="solid"
```

```
    filltransparency="0,7" fillcolor="153,0,0" boundarycaptype="round" />
  </SIMPLERENDERER>
     <COORDSYS id="31467" />
</LAYER>

<!-- BESCHRIFTUNG ALLER VERWALTUNGSSITZE  -->

<LAYER type="featureclass" name="Verwaltungssitz" visible="false"
  id="verwaltung" minscale="1:400000" maxscale="1:5000000">
  <DATASET name="verwaltung" type="point" workspace="shp_ws-100" />

     <GROUPRENDERER>
       <SIMPLERENDERER>
         <SIMPLEMARKERSYMBOL antialiasing="true" color="255,0,0" type="circle"
           width="6" outline="0,0,0"/>
       </SIMPLERENDERER>

       <SIMPLELABELRENDERER field="GN">

          <TEXTSYMBOL antialiasing="true" font="Arial" fontstyle="bold"
           fontsize="10" fontcolor="40,40,40" interval="8" glowing="255,255,255"/>
       </SIMPLELABELRENDERER>
     </GROUPRENDERER>
<COORDSYS id="31467" />
</LAYER>


<!--

<LAYER type="acetate" name="scalebar" visible="true" id="scalebar">
   <OBJECT units="pixel">
      <SCALEBAR fontcolor="0,0,0" coords="10 8" barcolor="255,255,255"
        bartransparency ="1,0" fontsize="12" screenlength="100" barwidth="5"
        mapunits="meters" antialiasing="true" scaleunits="meters"
        outline="255,255,255" />
   </OBJECT>
     <COORDSYS id="31467" />
</LAYER>

      -->

</MAP>

  </CONFIG>
</ARCXML>
```

Anhang C

Beilagen

Im Rücken dieser Diplomarbeit befindet sich ein Datenträger, der die folgenden Dateien beinhaltet:

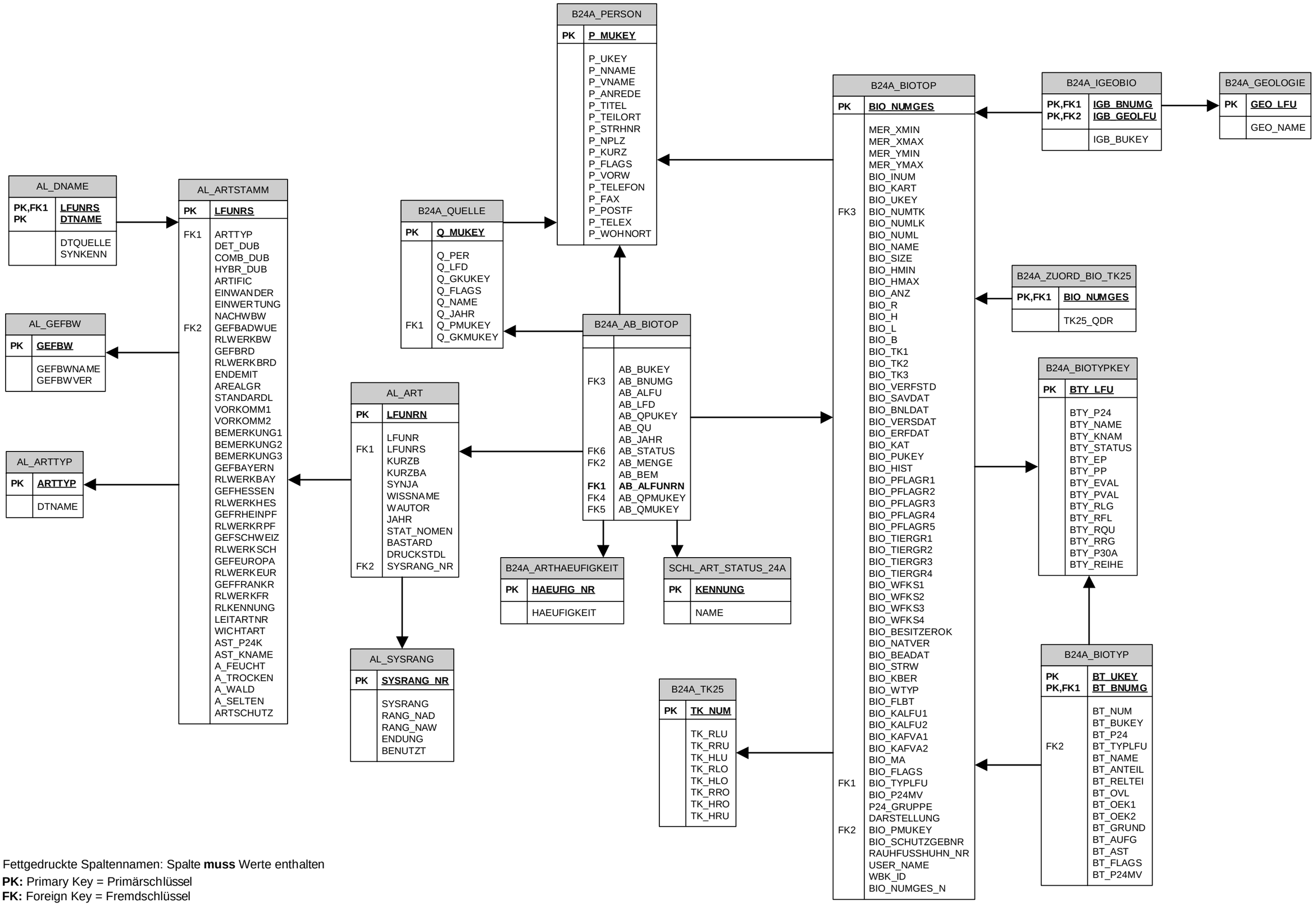
- Die Ausarbeitung der Arbeit im PDF-Format.
- Das Paket für die Installation des Konvertierungsprogramms zum Import der Daten der § 24a-Biotopkartierung in das Oracle Datenbankschema ARTIS
- Ausschnitt aus dem ER-Modell des Oracle Datenbankschemas ALBIS (nur die Tabellen, die in Zusammenhang mit der § 24a-Biotopkartierung stehen) und das ER-Modell des Schemas ARTIS.

Anhang D

Entity-Relationship-Diagramme

Aufgrund der Größe der ER-Diagramme der Schemata ALBIS und ARTIS wurden diese auf zwei DIN A3-Blättern ausgedruckt und auf DIN A4-Blattgröße gefaltet. Sie sind auf den nächsten beiden Seiten zu finden.

AUSSCHNITT AUS DEM DATENBANKSCHEMA ALBIS

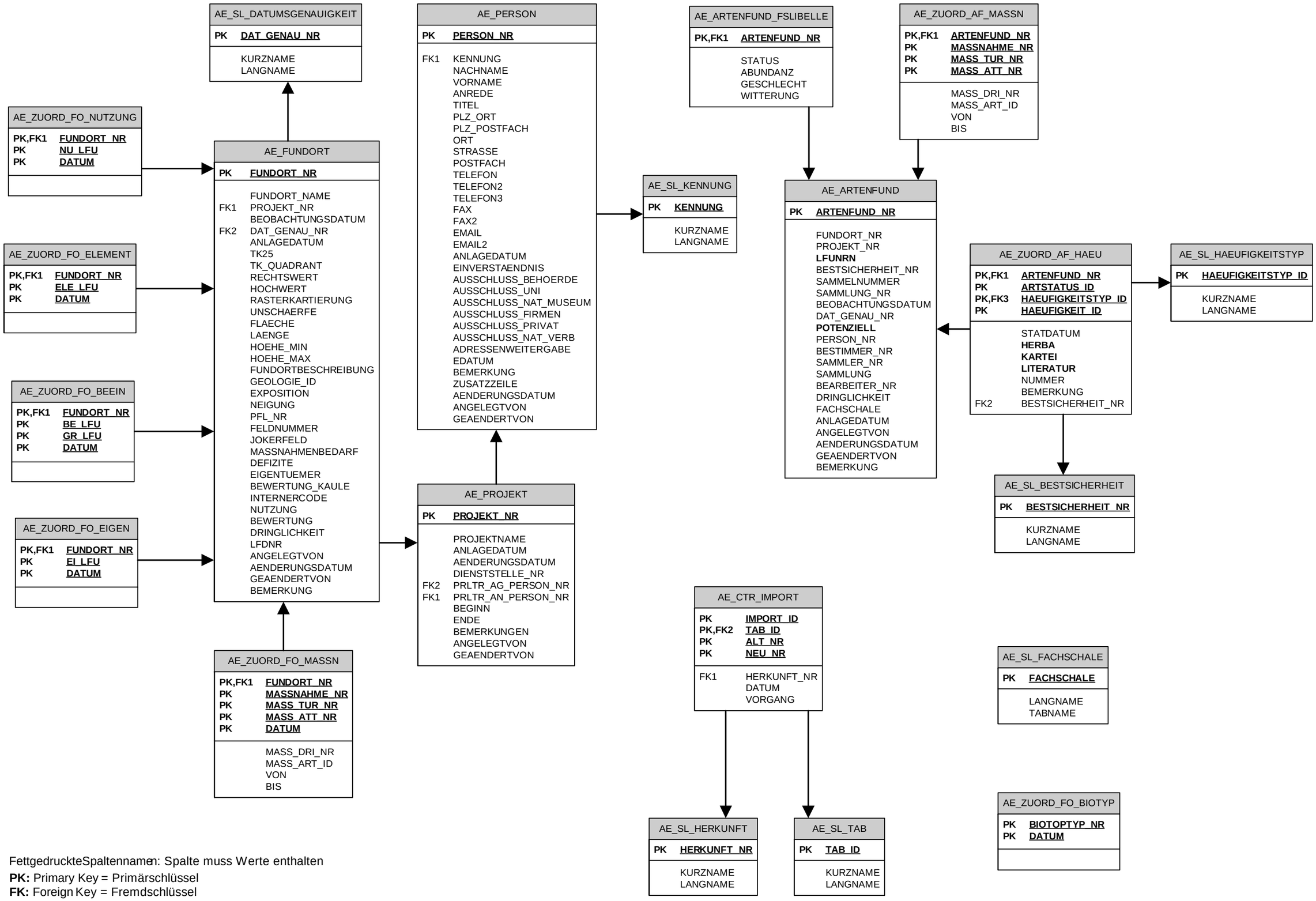


Fettgedruckte Spaltennamen: Spalte **muss** Werte enthalten

PK: Primary Key = Primärschlüssel

FK: Foreign Key = Fremdschlüssel

DAS DATENBANKSCHEMA ARTIS



FettgedruckteSpaltennamen: Spalte muss Werte enthalten

PK: Primary Key = Primärschlüssel

FK: Foreign Key = Fremdschlüssel

Literaturverzeichnis

[BG04] Bauer, A. (Hrsg.) ; Günzel, H. (Hrsg.): *Data Warehouse Systeme – Architektur, Entwicklung, Anwendung.* 2., überarbeitete und aktualisierte Auflage. dpunkt.verlag, 2004

[dis06a] disy Informationssysteme GmbH: *Aufbau einer BI-Lösung mit disy Cadenza.* 2006. – http://disy.net/plattform.html
Besucht am 19.10.2006

[dis06b] disy Informationssysteme GmbH: *disy Cadenza.* 2006. – http://disy.net/cadenza.html
Besucht am 19.10.2006

[ESR04a] ESRI: *ArcIMS 9-Architecture and Functionality – An ESRI White Paper.* 2004. – im Internet unter: http://support.esri.com/index.cfm?fa=knowledgebase.whitepapers.viewPaper&PID=16&MetaID=813 , Unterpunkt „ArcIMS“

[ESR04b] ESRI: *ArcIMS 9-ArcXML Programmer's Reference Guide.* 2004

[Her04] Hermann und Lenz Services GmbH: *TOAD Grundlagen.* Mai 2004

[Lan95] Landesanstalt für Umwelt, Messungen und Naturschutz Baden-Württemberg: *XfaWeb - Umwelt-Fachinformationen im WWW – Karten zu Natur und Umwelt.* Stand: 1995. – http://www.xfaweb.baden-wuerttemberg.de/nafaweb/berichte/plpw_01/karten0041.html
Besucht am 18.8.2006

[Lan04] Landesanstalt für Umwelt, Messungen und Naturschutz Baden-Württemberg: *XfaWeb - Umwelt-Fachinformationen im WWW- Info 3/2004.* Stand: Dezember 2004. – http://www.xfaweb.baden-wuerttemberg.de/nafaweb/
Besucht am 18.8.2006

[Lan06a] Landesanstalt für Umwelt, Messungen und Naturschutz Baden-Württemberg: *anw-lfu-nat.doc.* 2006. – Intranet: http://www.lubw.bwl.de
Besucht am 18.8.2006

[Lan06b] Landesanstalt für Umwelt, Messungen und Naturschutz Baden-Württemberg: *XfaWeb - Umwelt-Fachinformationen im WWW.* 2006. – http://rips-uis.lfu.baden-wuerttemberg.de/rips/natura2000/navigation/start.htm
Besucht am 18.8.2006

[Lan06c] Landesanstalt für Umwelt, Messungen und Naturschutz Baden-Württemberg: *XfaWeb - Umwelt-Fachinformationen im WWW.* 2006. –

http://www.xfaweb.baden-wuerttemberg.de/nafaweb/
Besucht am 18.8.2006

[Lon05] Loney, Kevin: *ORACLE Database 10g, Die umfassende Referenz*. 1. Auflage. Carl Hanser Verlag, 2005

[Man00] Mansfield, Richard: *Visual Basic für Dummies*. 1. Auflage. MITP-Verlag, 2000

[Mic06] Microsoft Corporation: *ADO API Reference*. 2006. –
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/
Besucht am 22.8.2006

[MLR06] Ministerium für Ernährung und Ländlichen Raum Baden-Württemberg: *Naturschutz in Baden-Württemberg*. 2006. –
http://www.naturschutz.landbw.de
Besucht am 18.08.2006

[Ope06] Open Source Project: *Community Mapbuilder*. 2006. –
http://communitymapbuilder.org/
Besucht am 23.10.2006

[Ora01] Oracle Corporation: *Introduction to Oracle9i: SQL, Student Guide*. Vol. 1. October 2001

[Ora05] Oracle Deutschland GmbH: *Oracle Warehouse Technologie – Single-Engine-Based-Data-Warehouse*. 2005. –
http://www.doag.de/pub/docs/sig/dwh/2005-11/Datenqualitaet_10_2005_Doag_Kurz.ppt
Besucht am 20.11.2006

[Ora06] Oracle Corporation: *Oracle Warehouse Builder User's Guide 10g Release 2 (10.2)*. Juni 2006

[Per98] Perry, Greg: *SAMS Teach Yourself Visual Basic 6 in 21 Days*. 1. Auflage. SAMS Verlag, 1998

[PHP05] PHP-Dokumentationsgruppe: *PHP Handbuch*. 2005. –
http://www.php-center.de/de-html-manual/
Besucht am 9.10.2006

[Reg00] Regionales Rechenzentrum für Niedersachsen/ Universität Hannover: *Visual Basic 6.0, Grundlagen*. 3., unveränderte Auflage. September 2000

[UIG04] Umweltinformationsgesetz (UIG): idF v. 22. Dezember 2004. –
im Internet unter:
http://bundesrecht.juris.de/bundesrecht

[UMN06] Plone Foundation et al.: *Heimatseite des MapServers*. 2006. –
http://mapserver.gis.umn.edu/
Besucht am 18.10.2006

[Uni03] Universität Rostock: *Geoinformatik Service – GI LEXIKON*. 2003. – http://www.geoinformatik.uni-rostock.de/lexikon.asp Besucht am 20.11.2006

[Uni06] Universität Regensburg: *FLOREIN - Programm zur Erfassung, Bearbeitung und Auswertung floristischer Daten*. 2006. – http://www.biologie.uni-regensburg.de/Botanik/Florkart/ Besucht am 18.8.2006

[Wik06a] Wikimedia Foundation Inc.: *Microsoft Visual Studio - Wikipedia, The Free Encyclopedia*. 2006. – http://en.wikipedia.org/wiki/Microsoft_Visual_Studio Besucht am 22.8.2006

[Wik06b] Wikimedia Foundation Inc.: *Visual Basic - Wikipedia, Die freie Enzyklopädie*. 2006. – http://de.wikipedia.org/wiki/Visual_Basic Besucht am 16.11.2006

Glossar

A

AbfaWeb Abfall-Fachinformationen im World Wide Web.

ADO ActiveX Data Objects. Schnittstelle, die auf ActiveX basiert und zum Zugriff auf Datenbanken verwendet wird.

AEP Artenerfassungsprogramm.

ALBIS ehemals Arten-, Landschafts- und Biotop-Informationssystem. Jetzt nur noch Oracle Datenbankschema.

AlfaWeb Altlasten-Fachinformationen im World Wide Web.

ALK Automatisierte Liegenschaftskarte.

API Application Programming Interface. Programmierschnittstelle, die von einer Software zur Verfügung gestellt wird und den Zugriff auf Datenbanken oder Hardware und die Erstellung einer grafischen Benutzeroberfläche ermöglicht.

ARTIS Arteninformationssystem.

ASP Artenschutzprogramm.

B

BofaWeb Bodenschutz-Fachinformationen im World Wide Web.

C

CGI Common Gateway Interface. Standard für den Datenaustausch zwischen Webserver und anderer Software, die Anfragen bearbeitet.

ChemfaWeb Behördliches Chemikalienmanagement in Baden-Württemberg.

D

DAD Datenaustauschdienst. Import und Export-Schnittstelle, die den automatisierten Austausch von Daten zwischen lokalen Datenbanken und der Referenzdatenbank ermöglicht.

DBI-Modul Database-Interface-Modul. Das Modul stellt Programmen, die in der Programmiersprache Perl geschrieben wurden, eine einheitliche Schnittstelle zu relationalen Datenbanken zur Verfügung.

F

FFH Fauna(=Tierwelt), Flora=(Pflanzenwelt), Habitat=(Lebensraum).

FIS-Natur Fachinformationssystem Natur.

FND Flächenhaftes Naturdenkmal. Schutzgebietskategorie, die naturschutzwürdige Flächen bis 5 ha Größe umfasst.

FofaWeb Umweltforschung in Baden-Württemberg.

G

GIF Graphics Interchange Format. Grafikformat, dass gute Komprimierung durch Verringerung der Farbtiefe auf maximal 256 Farben erlaubt.

GIS Geoinformationssystem.

GISterm Java-basiertes Geoinformationssystem des Ministeriums für Umwelt und Verkehr und der disy Informationssysteme GmbH.

H

HTTP Hypertext Transfer Protocol. Protokoll zur Übertragung von webbasierten Inhalten.

I

ITZ Informationstechnisches Zentrum. Abteilung 5 der Landesanstalt für Umwelt, Messungen und Naturschutz Baden-Württemberg.

J

JDK Java Development Kit. Entwicklungsumgebung der Programmiersprache Java.

JPEG Joint Photographic Experts Group. Grafikformat, das sowohl eine verlustfreie wie auch verlustbehaftete Komprimierung ermöglicht.

JRE Java Runtime Environment. Laufzeitumgebung für Java.

L

LSG Landschaftsschutzgebiet. Wird nach § 29 NatSchG zur Erhaltung der natürlichen Vielfalt, Eigenart und Schönheit der Landschaft ausgewiesen.

LUBW Landesanstalt für Umwelt, Messungen und Naturschutz Baden-Württemberg.

M

MLR Ministerium für Ernährung und Ländlichen Raum.

MNDNR Minnesota Department of Natural Resources.

N

NafaWeb Naturschutz-Fachinformationen im World Wide Web.

NAIS Naturschutzinformationssysteme.

NASA National Aeronautics and Space Administration. Zivile amerikanische Bundesbehörde für Luft- und Raumfahrt.

NSG Naturschutzgebiet. Gebiet, in dem ein besonderer Schutz von Natur und Landschaft aus naturgeschichtlichen, wissenschaftlichen, landeskundlichen oder kulturellen Gründen oder zur Erhaltung von Lebensgemeinschaften oder Biotopen bestimmter wildlebender Tier- und Pflanzenarten notwendig ist.

O

OCI Oracle Call Interface. Programmierschnittstelle zum Zugriff auf Oracle Datenbanken.

ODBC Open DataBase Connectivity. Standardisierte Datenbankschnittstelle auf Basis von SQL.

OGC Open Geospatial Consortium. Gemeinnützige Organisation, die versucht, die Entwicklung von raumbezogener Informationsverarbeitung (insbesondere von Geodaten) auf Basis allgemeingültiger Standards festzulegen.

OLE-DB Schnittstelle zur Anwendungsprogrammierung (API), die den Zugriff auf Datenquellen ermöglicht.

P

PDA Personal Digital Assistent. Kleiner tragbarer Computer, der u.a. zur Verwaltung von Terminen und Adressen dient. Häufig auch synonym zu Handheld oder Organizer verwendet.

PEPL Pflege- und Entwicklungspläne.

PNG Portable Network Graphics. Grafikformat, das eine hohe verlustfreie Komprimierung ermöglicht.

R

Repository zu deutsch Lager, Archiv, Vorrat. Begriff aus dem Software-Engineering. Bezeichnet grundsätzlich ein Datenhaltungssystem, in dem sämtliche Informationen über wiederverwendbare Softwarebausteine enthalten sind.

RIPS Räumliches Informations- und Planungssystem. Bestandteil des Umweltinformationssystems Baden- Württemberg.

S

SPA Special Protected Area. Gebiet aus dem Schutzgebietsverbundsystem NATURA 2000.

SQL Structured Query Language. Abfragesprache für relationale Datenbanken.

T

TK 25 Topographische Karte 1: 25.000.

U

UMN University of Minnesota.

URL Uniform Resource Locator. Häufig als Synonym für Webadresse verwendet.

W

WAP Wireless Application Protocol. Protokoll für die drahtlose Übertragung von Informationen, z.B. über ein Mobiltelefon.

WIBAS Behördenübergreifendes Informationssystem für die Bereiche Wasser, Immissionsschutz, Boden, Abfall und Arbeitsschutz.

WMS Web Map Service. Darunter versteht man die Internet-gestützte Erstellung von Karten innerhalb eines verteilten Geoinformationssystems.

X

XML eXtensible Markup Language. Metasprache, die zur Beschreibung von Dokumenten verwendet wird und vom World Wide Web Consortium (W3C) definiert wurde. Bezeichnet wie HTML eine Untermenge der Sprache SGML, die der Beschreibung von Webseiten dient.