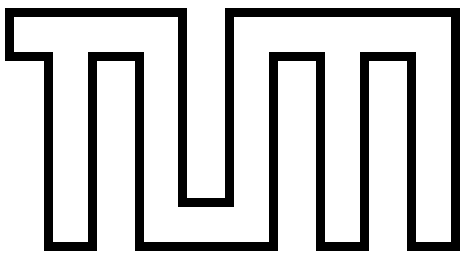


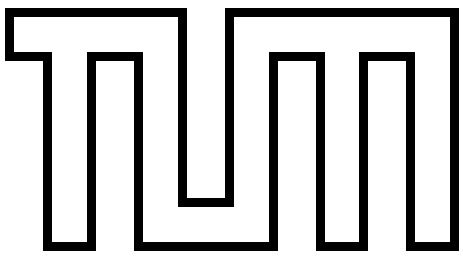
TECHNISCHE
UNIVERSITÄT
MÜNCHEN

INSTITUT FÜR INFORMATIK

Diplomarbeit

Integration des Managements des Domain-Name-Systems in die vorhandene Managementumgebung des Leibniz-Rechenzentrums (LRZ)

Heiko Hauck



TECHNISCHE
UNIVERSITÄT
MÜNCHEN

INSTITUT FÜR INFORMATIK

DIPLOMARBEIT

Integration des Managements des Domain-Name-Systems in die vorhandene Managementumgebung des Leibniz-Rechenzentrums (LRZ)

Bearbeiter: Heiko Hauck
Aufgabensteller: Prof. Dr. Heinz-Gerd Hegering
Betreuer: Dipl. Inform. Kirsten Heiler
Dr. Robert Valta
Abgabetermin: 15. November 1994

Erklärung

Hiermit erkläre ich, daß ich die vorliegende Arbeit selbständig angefertigt und nur die angegebenen Quellen und Hilfsmittel verwendet habe.

München, den 9. November 1994

(Heiko Hauck)

Inhaltsverzeichnis

Abbildungsverzeichnis

Kapitel 1

Einleitung

1.1 Motivation

Nach dem Bau des ersten Personalcomputers in den 70er Jahren kam es im darauffolgenden Jahrzehnt zum allgemeinen Durchbruch des Mikrocomputers. Seine Vertreter waren kaum noch aus Büros, Labors, Banken, aber auch privaten Wohnzimmern mehr wegzudenken.

Was diesen Systemen am Anfang aber weitgehend noch fehlte, war die Fähigkeit, miteinander zu kommunizieren. Es handelte sich oft um isolierte Systeme, bei denen die Daten von Hand eingegeben oder mittels zumeist magnetischer Datenträger geladen werden mußten. Dadurch war ein manuelles Eingreifen seitens des Anwenders bei Kommunikationsvorgängen erforderlich, was einen der wesentlichen Vorteile der elektronischen Datenverarbeitung, die (vollständige) Automatisierung von Arbeitsvorgängen, zwar nicht zunichte machte, aber dennoch deutlich einschränkte.

Um einen Datenaustausch zwischen Rechnern ohne umständlichen und langwierigen Transport eines phyischen Mediums (Band, Diskette, ...) zu ermöglichen, wurden Kommunikationsnetze geschaffen. Deren Bedeutung hat in den vergangenen Jahren enorm zugenommen. Auch wenn derzeit noch zwischen lokalen (LAN) und Weitverkehrsnetzen (WAN) unterschieden wird, so tragen moderne Hochgeschwindigkeitstechnologien wie ATM (Asynchronous Transfer Mode) zu deren Vermengung bei. Damit wird es möglich, die in letzter Zeit viel zitierten „Datenautobahnen“ zu realisieren, mit denen viele heute erst begrenzt mögli Multimedia-Anwendungen der breiten Masse zugänglich gemacht werden so da hierfür eine enorm hohe Bandbreite erforderlich ist.

Auf dem Weg dorthin müssen wir uns aber noch mit den Problemen exi der und erprobter Netze auseinandersetzen, um dann mit der daraus ge

Erfahrung Neues schaffen zu können. Zu den genannten Netzen gehört als einer der bekanntesten WAN-Vertreter das ARPA-Internet, das ursprünglich auf ein Forschungsprojekt des amerikanischen Verteidigungsministeriums (DoD) zurückgeht, sich in den vergangenen Jahren aber stark als Forschungs- und Hochschulnetz etablierte und in jüngster Zeit auch immer mehr für kommerzielle Zwecke genutzt wird.

Die vorliegende Arbeit beschäftigt sich mit dem Domain Name System (DNS), einer Anwendung auf dem Internet. Dabei handelt es sich um ein System, mit dessen Hilfe jedem Rechner im gesamten Netz ein Name zugeordnet wird, durch den er global eindeutig identifiziert werden kann. Damit wird die Verwendung umständlicher Adressen (Zahlen) überflüssig.

Derartige verteilte Anwendungen – das Domain Name System wird durch Rechner realisiert, die im ganzen Netz verteilt sind – sind aber sehr komplex und damit fehlerträchtig. Deshalb müssen Möglichkeiten geschaffen werden, das System „in den Griff“ zu bekommen, d. h. das System muß gemanaget werden.

Für das Management bieten sich einerseits isolierte, proprietäre und auf der ande
ren Seite integrierte Verfahren an. Während erstere öfters verfügbar und
sind, bietet integriertes Management weitaus mehr Sicherheit[1] und
ist deshalb vorzuziehen. Um letzteres zu realisieren, bieten sich
verschiedene Wege an:

- Anbindung isolierter Werkzeuge und Verfahren an ei
griertes Management. Hierzu sind Module zu entw
handenen Werkzeug aufsetzen und eine Schnittste

- Entwicklung neuer Werkzeuge und Verfahren,
nentplattform eingebunden werden. Dabei n
managenden Objektes berücksichtigt werde
tionen dem Manager an der Plattforma
bleiben sollen, sofern er sie nicht für

Die letztgenannte Forderung läßt sich
tems in Form einer Netzdokumentat
nen allgemein und unabhängig v
che Komponenten oder Anwendun
Plattform bzw. des (darin enthal
lich.

Am Leibniz-Rechenzentrum ist
teil der lokalen Management

[1] Sicherheit in Bezug auf K

Arbeit liegt nun darin, ein Verfahren zu entwickeln, das ein Werkzeug enthält, durch das das Domain Name System an die genannte Datenbank angebunden wird, und so sein Management ermöglicht wird. Davon ist in erster Linie das Konfigurationsmanagement betroffen, da die Datenbank u.a. sämtliche zur Konfiguration notwendigen Informationen bereitstellen soll.

1.2 Vorgehensweise

In Kapitel 2 wird zunächst eine allgemeine Einführung in das Domain Name System, den Gegenstand dieser Arbeit, gegeben. Dabei wird gezeigt, welches Problem zu lösen war, das letztlich zur Einführung des DNS führte, wie dieses vor und nach dessen Einführung gelöst wurde, und welche Vorteile das DNS mit sich brachte.

Mit dem anschließenden 3. Kapitel wird eine Beschreibung der Konfiguration des Domain Name Systems eingeschoben. Neben einer systematischen Unterteilung der gesamten vom DNS zur Verfügung gestellten Information wird anhand von kleinen Beispielkonfigurationsdateien aufgezeigt, wie man eine eigene DNS-Umgebung zum Laufen bringt.

Auch wenn deren Spezifikationen dem Netzverwalter letzlich verborgen bleiben soll (vgl. Abschnitt 1.1), so ist deren Kenntnis für die vorliegende Arbei erläßlich, um das neu zu implementierende Datenbankschema und Werkzeug stehen zu können. Die Dateien stellen die systemnäheste vom Netzverwal einflußbare Schicht dar.

Mit dem 4. Kapitel wird diesbezüglich ein Schnitt gemacht. Nach der des Managements allgemein und speziell für das Domain Name System sen Management aus Betreibersicht, d.h. in diesem Falle aus der Sic Administrators, präsentiert. Neben seinen Aufgaben und den damit Anforderungen werden Möglichkeiten aufgezeigt, die Aufgaben u allerdings isolierter Natur sind.

Deren Nachteile führen zum integrierten Management. Hier tungsweisende Ansätze präsentiert, einer aus dem Berei nents und ein anderer mit Hilfe eines Dokumentation Rechenzentrum u.a. als Netzdokumentationsdatenban nannte Ansatz wird bzgl. des Konfigurationsma verfolgt, weshalb am Schluß dieses Kapitels an die Managementumgebung stellen, genan

Die Umsetzung dieses Verfahrens ist Gege ne Netzdokumentationsdatenbank vorha abzulegen sind, muß man sich zunäch

Unabhängig davon werden die Anforderungen an das Design der neuen (zusätzli-
chen) Datenbank gestellt, bevor unter deren Berücksichtigung das Design selbst
entworfen wird. Anschließend wird versucht, die sich ergebenden neuen Relatio-
nen in die alten zu integrieren. Schließlich gilt es noch einige organisatori
Probleme zu lösen, bevor das Verfahren eingesetzt werden kann.

Das 6. Kapitel beschreibt schließlich die Implementierung des Verfah
zählt zumeinen das exakte Relationenschema mit einer Beschreib
Attribute und zumanderen das Werkzeug, an das wiederumeigen
gen gestellt werden.

Das Werkzeug wird in Formeines Perl-Skripts realisiert. E
beschreibung wird zunächst eine Übersicht präsentiert, b
Module, aus denen es sich zusammensetzt, eingegange
Demonstration der Ein- und Ausgabe des Programms r

Das 7. Kapitel gibt schließlich einen Ausblick über
zu tun ist. Dazu zählen zumeinen mögliche Erw
gen des implementierten Verfahrens und zumande
Managementplattform.

Kapitel 2

Einführung in das Domain System

Dieses Kapitel beschäftigt sich mit dem Domain Nam
nent in dieser Arbeit in eine vorhandene Mangement um
soll. Dabei handelt es sich um eine Anwendung aus
bekannte Anwendungen sind z.B. NIS (Network Infor
(Network File System).

Zunächst wird in Abschnitt 2.1 ein Problem vorge
sammenhang mit Kommunikationsnetzen auftritt.
Identifizierung der einzelnen teilnehmenden Kor
mit Zahlen arbeiten, ist es für den Menschen le

Ein Lösungsansatz zu diesem Problem, wie er v
Systems zum Einsatz kam, wird in Abschnitt 2.2
die Probleme dieses Verfahrens aufgezeigt,
gefunden werden mußte.

Hierbei handelt es sich um das Domain Nam
Überblick über die damit einhergehende I
schen Namensraum sowie die Art und
Informationen zu den jeweilgen Anwe

2.1 Motivation

Komponenten eines elektronischen
numerischen Wert identifiziert und a
Beispiel ist das Telefonnetz d
lefons durch Eingabe von

Auch Computernetze kennen Adressen, die je nach verwendetem Protokoll unterschiedlich ausfallen. So verwendet zum Beispiel das im LAN-Bereich sehr verbreitete Ethernet eine 48Bit-Adresse, die von den Herstellern der Adapterkarten bei deren Produktion eindeutig festgelegt wird. Damit ist dann sichergestellt, unter ein und derselben Adresse nur ein einziger Rechner erreichbar is Verwechslungen ausgeschlossen werden.

Im ARPA-Internet vernetzte Rechner verwenden das IP-Protokoll. Die gen IP-Adressen haben einen Umfang von 32 Bit und werden in der durch Punkte getrennte dezimale Bytewerte dargestellt. So besi spiel der Rechner, auf dem gerade dieser Text eingegeben wird **129. 187. 10. 20**.

Numerische Adressen werden eingesetzt, da sie vom Computer beitet werden können. Dagegen sind sie für den Benutzer da der Mensch im allgemeinen ein besseres Namens- wi sitzt. Um den Bedürfnissen des Anwenders entgegenz deswegen entschlossen, für Rechner auch alphanumeris (Beispiel: **sun1**).

Damit ein solcher Service angeboten werden kann, i dig, der die Namen auf die numerischen Adressen ab wird als Namensauflösung (Address Resolution) b tinen im Betriebssystem der Rechner als Reso

Ein weiterer Vorteil dieser Abbildung bes Rechner erhalten bleiben können, wenn s weise beim Umzug eines Rechners in

2.2 Früheres Verfah

Um die Address Resolution durchf *HOSTS. TXT* kreiert [Sant 93], IP-Adressen enthielt. Diese Network Information Cente sein, auf dem unter Verwend bindung zu einem anderen Re teilungsprozeß wurde mit (siehe Abbildung 2.1).

Lokal wurde die Datei ten dann die ber nehmen.

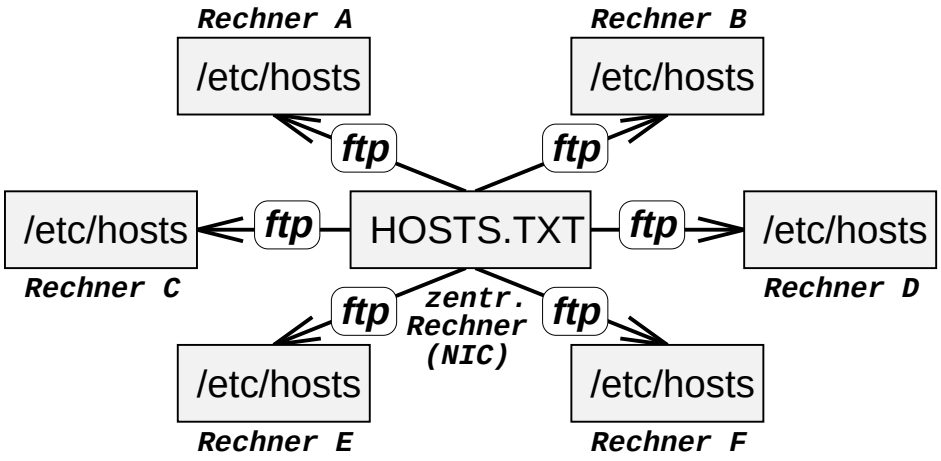


Abbildung 2.1: Verteilung von *HOSTS.TXT*

Dieses Verfahren hatte aber entscheidende Nachteile [Geor 93]:

- *Hohe Netzlast durch Verteilungsprozeß*
  Durch den Verteilungsprozeß wurde das Netz beträchtlich belaste
  sonders ungünstig war die Tatsache zu werten, daß die Netzl
  dratischer Komplexität $O(n^2)$ in Bezug auf die Zahl der Re
  bei einer Verdoppelung der Anzahl der Rechner die Datei
  so oft über das Netz geschickt werden mußte, sondern s
  verdoppelte.

- *Langsame Registrierung von Modifika*
  Alle Daten wurden bei diesem Ve
  verwaltet. Das bedeutete, daß
  werden mußte und erst nach e
  ganzen Netz bekannt wurd
  Internet (im Jahre 1993 c
  daß der notwendige
  Zeit erledigt werd

- *Kollisione*

  b

2.3 Das Domain Name System

Um den oben genannten Problemen zu entgegnen, wurde im Jahre 1984 das Domain Name System (DNS) eingeführt ([RFC 881], [RFC 882] und [RFC 883]). Dabei handelt es sich um einen Verzeichnisdienst, bei dem die Informationen nicht zentral, sondern verteilt gespeichert werden, mit einem hierarchischen statt flachen Namensraum.

Auf diese Punkte sowie auf das Verfahren, wie diese Information schließlich dort verfügbar gemacht wird, wo sie benötigt wird (Anwendung), wird in den folgenden Abschnitten eingegangen. Dabei wird auch der sehr wichtige Begriff des Nameservers erläutert.

2.3.1 Verteilung der Information

Beim DNS liegt ein über das ganze Netz verteiltes Datenbanksystem vor. Die Information, welcher Name auf welche Adresse abzubilden ist, wird (in der Regel) von derjenigen Organisation verwaltet und den anderen zur Verfügung gestellt, bei der sie auch tatsächlich anfällt. Dadurch wird das zweite Problem, die gleichzeitige Registrierung von Änderungen der DNS-Daten, gelöst.

Beispielsweise verwaltet das Leibniz-Rechenzentrum (LRZ) sämtliche relevanten Informationen bezüglich der eigenen Rechner selbst. Darüber hinaus macht das LRZ aber auch diesen Dienst für einen Teil der Bayerischen Hochschulen. Es ist also möglich, diese Aufgabe an andere zu übertragen, die sich nicht damit beschäftigen kann bzw. will.

2.3.2 Hierarchischer Namensraum

Das Problem der Namenskollisionen läßt sich dadurch eliminieren, indem Rechnernamen nicht flach, sondern in einer Baumstruktur hierachisch angeordnet werden (Teilbaum siehe Abbildung 2.2).

Die Namen setzen sich aus dem Hostnamen, also dem Namen des Rechners (zum Beispiel: **sun1**), und dem Domainnamen, der angibt, in welcher Institution (Bereich) der Rechner gehört (im Beispiel: **lrz-muenchen.de**) zusammen. Die Kombination von beidem wird als Fully Qualified Domain Name (FQDN) bezeichnet (im Beispiel also: **sun1.lrz-muenchen.de**).

Die Domainnamen setzen sich aus den Namen der Knoten im Pfad zusammen. Der Domainname eines Knotens besteht aus dessen Name sowie den Namen der im selben Pfad darüberliegenden Knoten – jeweils durch ein Punkt getrennt (Beispiel: **nm.informatik.uni-muenchen.de**). Der

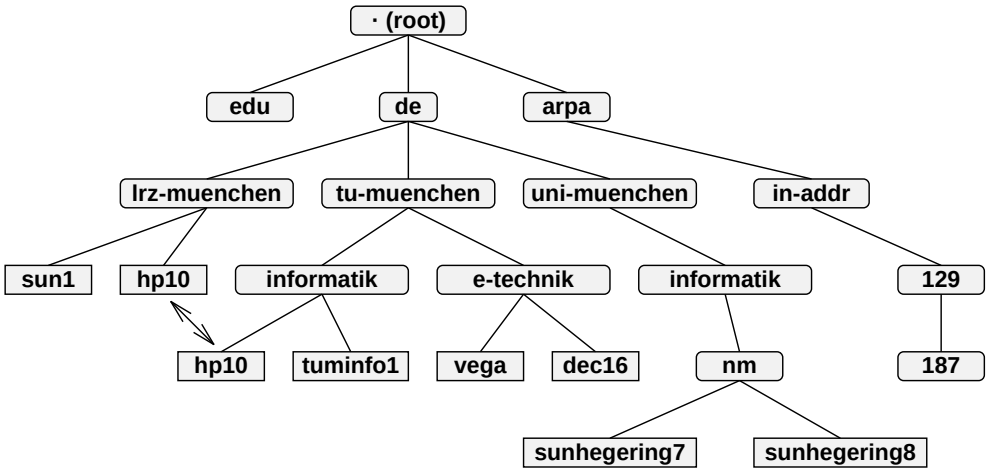


Abbildung 2.2: Hierarchischer Namensraum (Teilbaum)

die Root-Domain (**.**) dar, die darunterliegenden heißen Top-Level-Domains (**edu, de, ...**), die der nächstniedrigeren Schicht entsprechend Second-Level-Domains (**lrz-muenchen.de, tu-muenchen.de, ...**). Modalitäten zur Namensvergabe sind in [RFC 1032] beschrieben.

Alle Teilbäume eines Knotens bilden zusammen mit diesem eine Domain, der die darunterliegenden Subdomains und Rechner angehören. Zur Informationaufteilung (verteiltes Datenbanksystem) wird der Baum in Zonen eingeteilt, die von den einzelnen Institutionen in Eigenverantwortung verwaltet werden. Eine Zone umfaßt eine (Sub-)Domain abzüglich der delegierten (in einen anderen Verantworungsbereich übertragenen) Subdomains, die wiederum einer eigenen angehören.

Die Blätter im Baum stellen die Rechner mit ihren Namen, also den Hostnamen dar. Darin ist dann auch die entsprechende IP-Adresse (gegebenenfalls mehrere) gespeichert. Wie am Beispiel des Hostnamen **hp10** zu sehen ist es jetzt auch möglich, ein und denselben Hostnamen[1] mehrmals zu verwenden, wenn der Rechner einer anderen Domain angehört. Dasselbe gilt auch innerhalb zweier verschiedener Domainnamen (in der Abbildung: **informatik.muenchen.de** $\neq$ **informatik.uni-muenchen.de**).

Eine Besonderheit stellt die Domain **in-addr.arpa** dar. Hier ist eine inverse Namensabbildung (Reverse Mapping): IP-Adresse $\rightarrow$ Name vorzunehmen. Dazu wird die IP-Adresse als dezimale Byte

[1] **hp10.informatik.tu-muenchen.de** ist ein Aliasname für **hp10.informatik.uni-muenchen.de**. Aliasnamen und Hostnamen sind aber gleichbedeutend, weshalb dies als real existierendes Beispiel gewählt wurde.

Nullen und Leerzeichen in umgekehrter Reichenfolge geschrieben, um den Postfix **in-addr.arpa** erweitert und als Name verwendet (Beispiel: **20.10.187.129.in-addr.arpa**). In dem damit referenzierten Blatt findet man dann einen entsprechenden Eintrag, der den Namen enthält (im Beispiel: **sun1.lrz.muenchen.**
Der in der Abbildung abgeschnittene Teilbaum enthält alle Einträge für ei
B-Netz (**129.187.␣␣␣.␣␣␣**) des LRZ.

Neben dem Forward Mapping (Abbildung: Name →IP-Adresse) und der eben behandelten Reverse Mapping bietet das Domain Name System noch weite Dienste/Informationen an. Der wichtigste ist das Mail-Routing [RFC974]. ben können aber auch Informationen über CPU und Betriebssystem der Rec abgefragt werden, sofern sie bereitgestellt werden. Weitere Information ben (derzeit) eine untergeordnete Bedeutung und sollen deshalb a Stelle nicht erwähnt werden.

2.3.3 Zugriff einer Anwendung auf DNS-Informationen

Im folgenden wird gezeigt, wie eine Anwendung (Programm) auf Informationen, die das Domain Name System zur Verfügung stellt, zugreiffen kann. Abbildung 2.3 gibt eine Übersicht über ein mögliches Szenario.

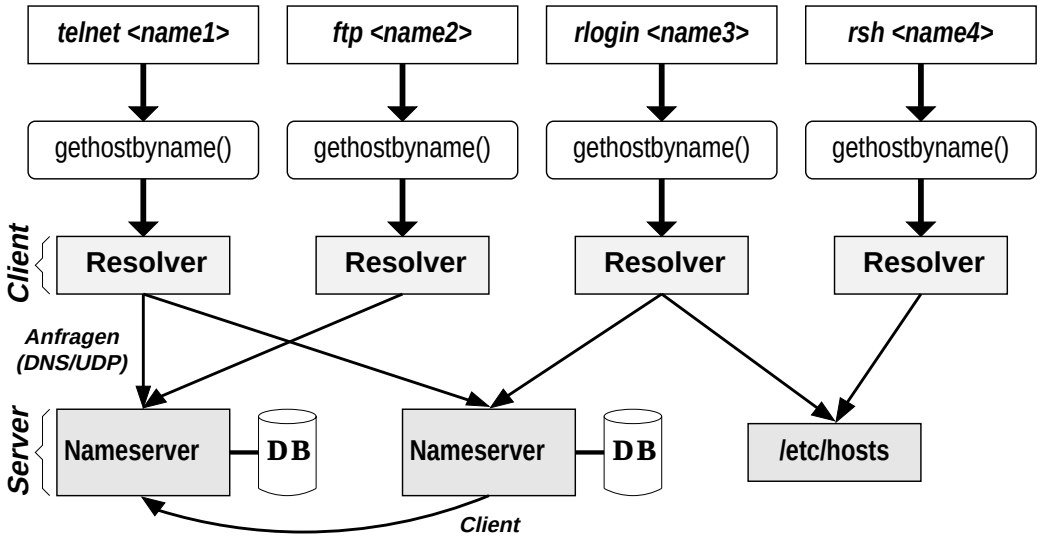


Abbildung 2.3: Informationsfluß (DNS-Anfragen)

Den Anwendungsprotokollen (telnet, ftp, ...) wird anstelle der IP-Adresse d Hostname bzw. der Fully Qualified Domain Name (FQDN) übergeben. Wird nur

der Hostname angegeben[2], so wird er um den Domainnamen erweitert. Das aufgerufene Programm bedient sich beim Verbindungsaufbau der Bibliotheksfunktion gethostbyname(). Damit werden die Resolverroutinen des Betriebssystems aufgerufen.

Die Resolverroutinen schicken die Anfragen an den/die nächstgelegenen Nameserver[3]. Je nach Konfiguration der Resolver (siehe Manpage *gethostent* [UCBK]) kann die Anfrage auch an mehrere Nameserver geschickt oder können vorhandene Einträge in */etc/hosts* zur Namensauflösung verwendet werden.

Ist der Nameserver nicht in der Lage, eine Anfrage selbst zu beantworten dern nur auf einen diesbezüglich kompetenteren Nameserver zu verweisen die Anfrage an letzteren geschickt werden. Diese evtl. rekursive Abf kann der Resolver dem Anwendungsprogramm abnehmen[4].

Ein Nameserver ist ein Prozeß, der Anfragen beantwortet, die er DNS-Protokolls [RFC 1035] zumeist als Datagramme (UDP/IP) er Zu diesem Zweck kann auch eine TCP/IP-Verbindung aufgebaut wer Nameserver läuft als Hintergrundprozeß (Daemon) auf ausgewählten eines Instituts oder einer Organisation.

Als eigene Informationsquelle besitzt er eine interne Datenban Konfigurationsdateien geladen wird. Daneben besteht aber Teile einer Datenbank anderer Nameserver zu kopieren, formationen daraus liefern zu können. Für diesen F dann als Secondary Nameserver, wohingegen er fü formation aus lokalen Konfigurationsdateien Nameserver agiert. Diese Spiegelung is ry Nameserver ausfallen kann und zuma Nameserver eine geringere Netzlast un satz gilt, daß für jede Zone mindest [Quar 94].

Daneben speichert ein Nameserver daß die gleiche Anfrage erne hat. Dieses als Caching server (Cache Nameserv anderen Nameservern deren Nameserver

[2] Dies wird in der P
Manpage *resolver* [UCBK, Option RES_DEFNAMES).
[3] Beim Domain Name System wird eine Client/Server Architektur eingesetzt. Der T
port der Information zwischen dem Nameserver (Informationsquelle) und dem Anwendungs-
programm (Client, Informationssenke) erfolgt über das Internet.
e Manpage *resolver* [UCBK, Option RES_RECURSE
Option RES_USEVC

An dieser Stelle muß darauf hingewiesen werden, daß die meisten Nameserver sowohl als Primary und Secondary Nameserver arbeiten (jeweils für eine oder mehrere verschiedene Zonen) als auch als Cache Nameserver für die Root-Domain. Letztere dient dazu, im Baum an Zweige zu gelangen, die nicht zu den selbst verwalteten gehören.

Kapitel 3

Konfiguration einer DNS-Umgebung

Beim Domain Name System handelt es sich um einen Ver
jenigen Informationen zur Verfügung stellt, die de
rationsvorgang mitgegeben werden. Dies geschieht
dateien.

Wegen ihres großen Umfangs müssen die Inform
Unterteilung erfolgt in zwei verschiedenen
schnitt 3.1 vorgestellt.

In Abschnitt 3.2 werden die Konfigurat
den Nameservern verwalteten Zonen s
spielen, präsentiert. Deren Kennt
Arbeit zu erstellendes Datenba
deren Hilfe entsprechende Date

Die Konfigurationsdateien
tüchtige DNS-Umgebung z
nuellen Konfiguration lieg
notwendig sind, sonde
stehen schwerwiegen
und dort eingezi

3.1 Unte

Die Unterteilu
mensionen

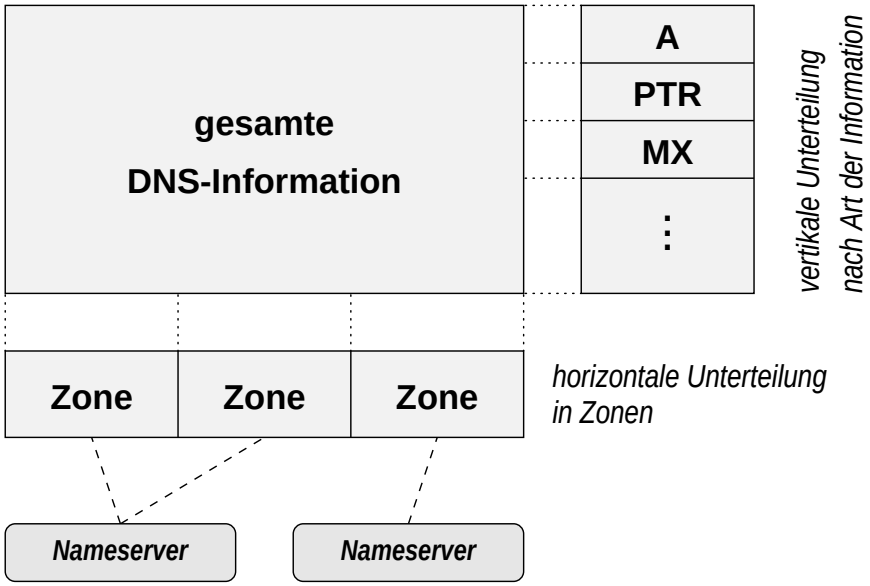


Abbildung 3.1: Unterteilung der DNS-Information

1. *Horizontale Unterteilung in Zonen*
   Bei einer verteilten An
   st

Tabelle 3.1 aufgeführt.

Typ	Bedeutung
A	Name → IP-Adresse (Foreward Map
PTR	IP-Adresse → Name (Re
MX	Mail Routi
HINFO	

Wie aus Abbildung 3.2 ersichtlich ist, teilen sich die f
wendigen Konfigurationsda

```
directory   /etc/namedb
; type      domain                  source/host    bak-file
cache       .                       named.cache
primary     lrz-muenchen.de         named.hosts
primary     128.129.in-addr.arpa    named.rev
primary     0.0.127.in-addr.arpa    named.local
secondary   eg                      193.227.1.1    eg.BF
secondary   gov.eg                  193.227.1.1    gov.eg.BF
```

3.2.2 Zonenkonfiguration

In den Zonendateien werden die Informationen festgelegt, d
Systemfür die jeweilige Zone zur
Da

```
ppp             IN NS    dfvgate
                IN NS    hp10
@               IN MX    140 cd1
localhost       IN A     127.0.0.1
sun1            IN A     129.187.10.20
sun2            IN A     129.187.10.21
dfvgate         IN A     129.187.10.25
hp10            IN A     129.187.13.22
cd1             IN A     129.187.13.3
```

- *named.rev*
  Bei der folgenden Datei Beispiel für das Reverse
  Netz (**129. 187.** ␣␣␣. ␣␣␣, un

```
$ORIGIN  187.1
@


20.10
20.11
195.15
```

- *named.loca*
  Die fo

Diese Datei kann je nachdem, ob auf demjeweiligen Rechner
installiert ist oder nicht, unters

- *Mit lo*

Kapitel 4

Anforderungen an das Management eines Nameserver

Dieses

4.1 Bedeutung des Managements

Bei modernen Rechnersystemen und Rechnernetzen genügt es nicht, sic
die Aufgaben, die sie zu erbringen haben,
eine Buchung

4.2 Notwendigkeit des Managements bzgl. des Domain Name Systems

m vorangehenden Kapitel wurde ein einfaches Beispiel gezeigt, wie sich ei
DNS-Umgebung konfigurieren läßt. Dabei wurden e
wichtige Zonen berücksic

Neben Problemen, mit denen der Anwender konfroniert wird, gibt es auch noc
andere, die z. B. die Betreiber von Nameservern unt
Abstimmung bei der Z

Dagegen werden im folgenden Aufgaben genannt, die in die *Betriebsphase* überführen bzw. in dieser von Bedeutung sind:

- Bereitstellen und Aktivieren
- Anpassung de

- Beim Bereitstellen von Nameservern ist festzulegen, welche Dienste j
einzelne anzubieten hat. In der Regel verf
Service für die i

Δt_g: Restgültigkeitsdauer eines gespiegelten Datenbestandes.
Erst nach Ablauf dieser Zeit fragt ein Se
zuständigen Primar

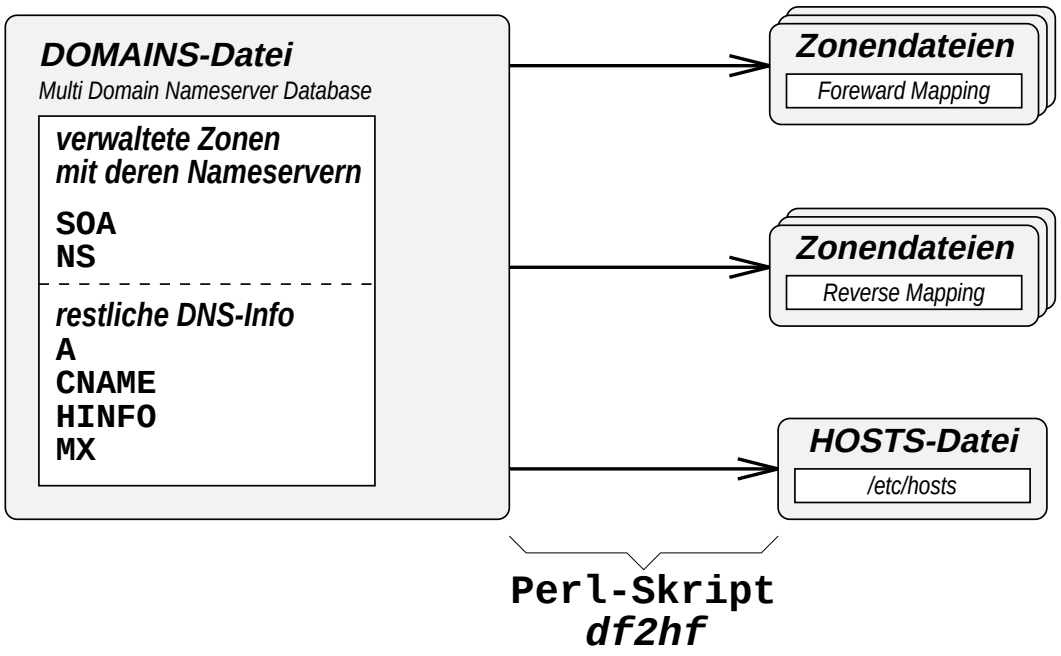


Abbildung 4.1: Einsatz des Generierungsskripts *df2hf*

In der vom DNS-Administrator zu erstellenden DOMAINS-
DNS-Informationen (Records) in
in Kapi

direkt voneinander abhängen, kann man sich doppelte Arbeit sparen.
Skript erstellt die PTR-Records automatisch
digen Zonen angib

- *Unterstützung von nur einem Nameserver*
  Aus der DOMAINS-Datei werden nur Zonendateien für einen Nameser
  generiert. Will man zwei oder mehrere Names
  ver für darin enthal

dnswalk: Werkzeug, das von einzelnen Domains Zonentransfers durchführt
die Ergebnisse auf interne Konsistenz und S

dnsstats: Werkzeug zur Analy

einemheterogenen Netz, da die eingesetzten Nameserver, Protokolle und Sch
stellen jeweils homogen sind, sondern viel me
Management verfah

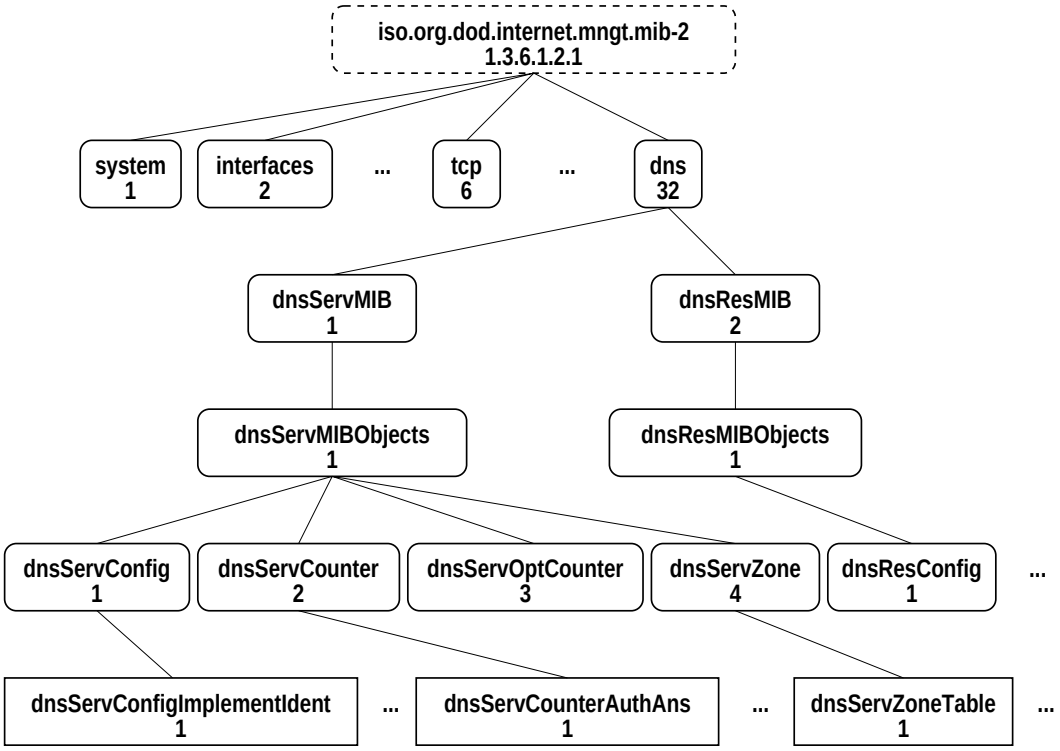


Abbildung 4.2: DNS in der MIB-2 des Internet-Registrierungsbaume

dnsServCounter: Statistische Angaben über Anfragen,
server gestellt wurden. Die
Anzahl de

Domain Name Systemdann im Rahmen des Internet-Managements verwaltet werden.

4.4.2 Abbindung des DNS an ein Dokumentationssystem

Ein Dokumentationssystemdient der Erfassung, Darstellung und Verwalt
physischen Infrastruktur eines Kommunikati
ist vor allemda

I n t e g r a t i o n[7], a u f d e r e n G r u n d l a g e d a n n e i n V e r f a h r e n z u m DNS-Konfigurations
m a n a g e m e n t e r a r b e i t e t w i r d .

Daneben k ö n n e n d a n n a u c h a n d e r e Anwend
e i n T r o u b l e T i c k e

In den Einträgen der Zonenkonfigurationsdateien sind Zeitinformationen an
geben, wie lange die jeweiligen Einträge gül
lange sie da

Wünschenswert wäre es, wenn dem DNS-Administrator ein Werkzeug (evtl. innerhalb einer Managament plattform) zur Verfügung
hand von Statistiken gut

interaktives Abfrage- und Manipulationswerkzeug von ORACLE) unterbleibt
sonst Inkonsistenzen zwischen der externen Dat
den Konfigurationsda

Kapitel 5

DNS-Konfiguration mit de
Netzdokumentati

Im folgenden

Im folgenden Abschnitt 5.3 wird ein Entity/Relationship Modell präsenti
das aus den Anforderungen entwickelte Desi
Relationensche

ie Abbildung 5.1[1] gibt einen Überblick über die Informationstabellen. Die e
gezeichneten Pfeile stellen die Beziehungen z
(Fremdschlüssel

das Domain Name Systems sind nur solche interessant, die über Schnitts
len mit IP-Protokoll verfügen:

```
select distinct kl.klasse, kl.name
from komponentenklasse kl, komponente ko
where kl.klasse=ko.klasse and
      ko.name=lp.komponente and lp.protc
```

Als Ergebnis dieser Anfrage er

```
KL NAM
-- ---
AS ATM
BI Bri
BR Bro
DI CDC
DR Dru
FK FDD
MC Mac
PC PC
PL Plo
RE Rec
RO Rou
SK Ste
TH Twi
TS son
UV X.2
WS Wor
XW XWi
```

Dabei

(nützlich für Joins, da dann nur jeweils ein Vergleichsattribut zu bearb
ten ist) als auch komponentenweise (wichtig
Vergleich stat

überwacht werden.

Ein entscheidendes Problembei Neuentwicklungen und Erweiteru
Produkten in Bezug auf die Akzeptanz de
formatik stellt

sowie der Gerätenummer (5. bis 8. Zeichen) zusammengesetzte Primärschlüs
NAME der Relation KOMPONENTE.

Nun ist ein weiteres Attribut (KLASSE) de
mit der des obi

5.2.1 Hosts

Unter „Host“ wird im folgenden eine Rechnereinheit verstanden, die *genau einen* Namen (Hostnamen) besitzt[5].

Anforderungen, die sich an das Objekt „Ho

Zonenzuordnung:
Jeder Host *mu*

Anforderungen, die sich an das Objekt „Zone“ stellen:

Vaterzone:
Zu jeder Zone (auch die delegierten fremdverwa
explizit die Vaterzone a

Mit dem Reverse Mapping eines Internet Class-A Netzes ergibt sich bei Ver
dung einer 16 Bit Subnetzmaske auch ein Beispie
Delegation der Zon
dem Byte we

Anforderungen, die sich an das Objekt „Nameserver“ stellen:

Lokaler Host:
Ein lokaler[10] Nameserver *muß* auf einem lokale
keinen Sinn, einen Nam

Verfügung stellen bzw. als Slave Nameserver arbeiten, ein sehr viel ger
Managementaufwand vonnöten ist, was die K
Überlegung ans

5.3 Das Datenbankdesign

5.3.1 Entity/Relationship Modell

bildung 5.3 zeigt das unter Berücksichtigung der genannten Anforderungen
entwickelte Entity/Relationship Modell.

Zentraler Gegenstand des Modells s
wesent

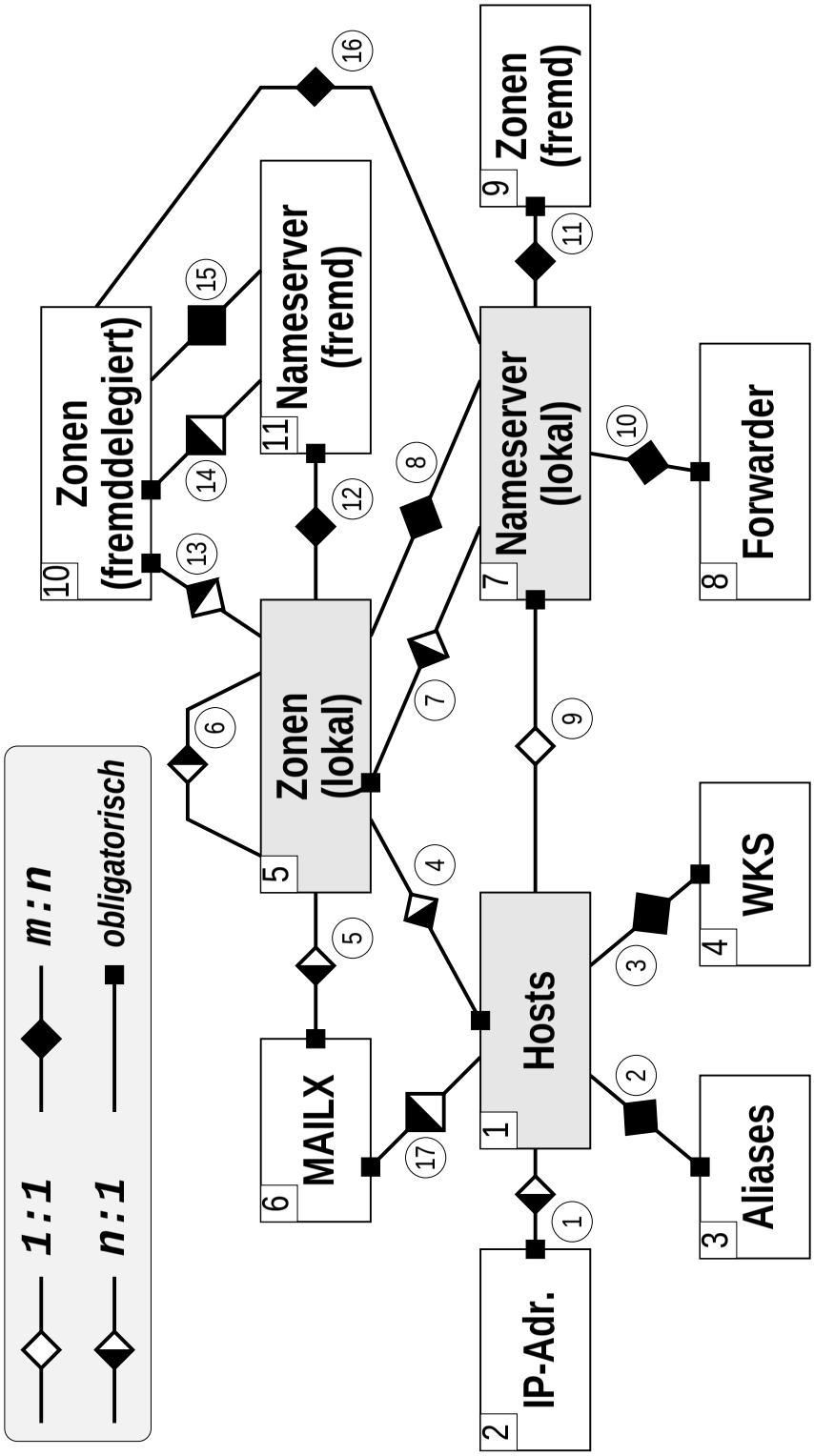


Abbildung 5.3: Entity/Relationsship Modell

3. ALIAS - NAMEN
Zu einem Host gehörige Alias-Namen. Hierfür ist eine Tabelle
zu erstellen.

4. WELL KNOWN SERVICES
Von einem Host angebotene
ei

5.3.3 Die Beziehungen (Relationships) und deren Integration in die Tabellen

1. IP-ADRESSEN ↔ HOSTS (**N:1**)
   Jeder Host hat eine oder mehrere IP-Adressen. Die Tabe
   um den Primärschlüssel der Tabe

2. ALI A

eine eigene Tabelle (SECONDARY_LL) zu erstellen. Diese enthält di
märschlüssel der Tabellen HOSTS und NAME

9. Lokale Nameserve
Jeder lok

16. FREMDE ZONEN (DELEGIERT) ↔ LOKALE NAMESERVER (SECONDARY SERVICE) (M: N)

Fremde, delegierte Zonen können von einem oder mehreren lokalen (Secondary) Nameservern ges

Hierfür soll die Tabelle LOGPORT (alt) herangezogen werden, die u.a. die Adressen enthält, was aber zu Problemen führt,

```
SQL> desc logport
 Name                           Null?    Type
 ------------------------------ -------- ----
 KOMPONENTE                              CHAR(8)
 PORT                                    CHAR(10)
 PROTOKOLL                      NOT NULL CHAR(3)
 ADRESSE                        NOT NULL CHAR(50)
 BEMERKUNG                               CHAR(35)
```

Die vier Attribute KOMPONENTE, PORT, PROTOKOLL und BEMERKUNG
sind zusammen nicht eindeutig, da eine Komponente mehrere logi
demgleichen Protokoll haben kann, u
meistens unbelegt

Die

Als Ergebnis aller bisher genannten Anbindungsversuche der Tabelle HO
die beiden vorhandenen Tabellen läßt sich fes
bindung geben

Während das erste Problem das Verfahren nur intern betrifft und im folg
den durch einen Kompromiß gelöst wird, muß sich
der Benutzer- und Syste

Weitere Forderungen, die sich im Zusammenhang mit der Eingabereihenfol
geben, bestehen darin, daß z.B. die IP-Adres
Services nach den

5.5.3 Zugriff auf Nameserver-Prozeß

Ein Problem, das mit der erfolgreichen Verteilung der Konfigurationsdate
nicht gelöst ist, ist die Prozeßsteuerung
von Signale

Kapitel 6

Implementierung

Dieses Kapitel beschäftigt sich schließli
bankrelatio

6.1 Aufbau der Relationen

Im vorangehenden Kapitel wurden mit Hilfe eines Entity/Relationsh
die notwendigen DB-Tabellen hergeleitet
Herleitung

6.1.1 Modifizierte Tabelle

Die in der Netzdokumentationsdatenbank vorhandene Tabelle INSTITU
wie folgt erweitert:

	Attributname	Wertebereich	NOT
1	id	char(4)	
2	organisation	ch	
3	name		
4	che		

3. Subdomainname, falls der Host einer Subdomain innerhalb sein
angehört. [optional]

4. Zone, der der Host angehör
tung: Di

- WKS

	Attributname	Wertebereich	NOT NULL
1	host	number(5,0)	×
2	service	char(240)	×

1. Host (Fremdschlüssel auf HOSTS.ID).
2. Angabe von IP-Adresse,

5. Name der zu erstellenden Zonendatei. [optional, standardmä
mainname + Suffix]

6. E-Mail Adresse des DNS-Adminis

7. Klasse d

Mit der folgenden Anweisung wird die Eindeutigkeit des Primärsch
gewährleistet:

```
create unique index del_zones_id on del_zo
```

- DEL_ZONES_SEC

	Attributname	Wertebereic
1	del_zone	numb
2	sns_ip_na	
3	s	

Mail Routing (über MX-RRs) wird im Rahmen dieser Arbeit nur in d
angegebenen Form unterstützt (siehe Kapitel

6.1.4 Nameserver-Tabellen

- NAMESERVERS

	Attributname	Wer
1	id	
2	ho	

7. Seriennummer der SOA-RRs des Loopback Interfaces (32Bit-Wert
ne VZ, siehe [RFC1035], Kapitel 3.3.13).
lation von DNS-I

- SECONDARY_LL (lokale Nameserver für lokale Zonen)

	Attributname	Wertebereich	NOT NULL
1	zone	number(5,0)	×
2	nameserver	number(5,0)	
3	backupfile	char(64	

1. Zone (Fremds

2.

- FORWARDERS

	Attributname	Wertebereich	NOT NULL
1	nameserver	number(5,0)	×
2	ip_addr	char(15)	×

1. Nameserver (Fremdschlüssel auf NAMESE
2. IP-Adresse des Master N
te

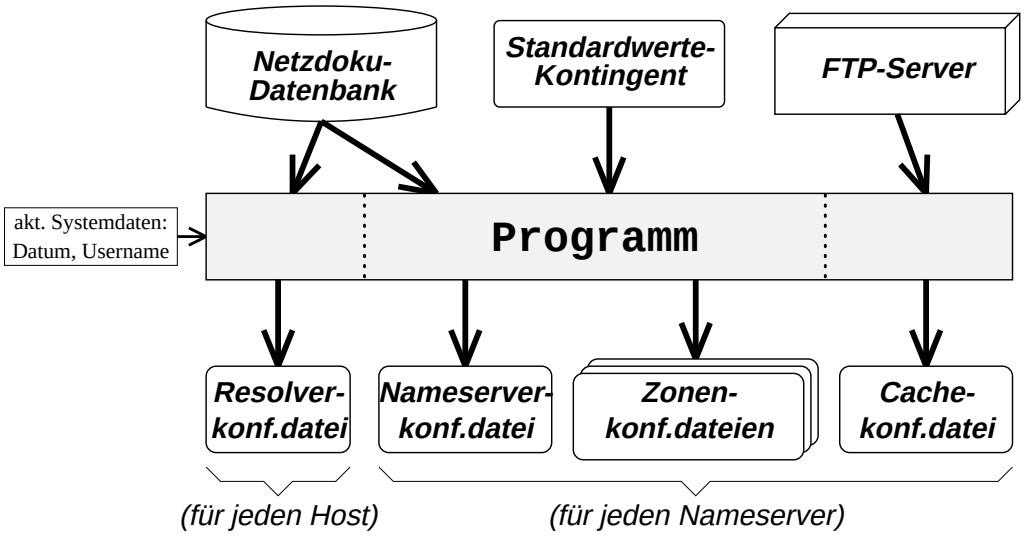


Abbildung 6.1: Informationsquellen und -senken

- Ein Kontingent an Standardwerten, die in der R
  den, wenn die entspre

6.2.2 Flexibilität und Komfort

In vielen DNS-Managementumgebungen liegt ein sehr umfangreicher Dat
stand vor, der beim Einsatz eines neuen Konfigu
lich gemacht wer

3. Ist dieses ebenfalls unbelegt, so wird der in der Konfigurationsd
letztgenannte Wert verwendet (NAI

4. Ist auch in der Konf

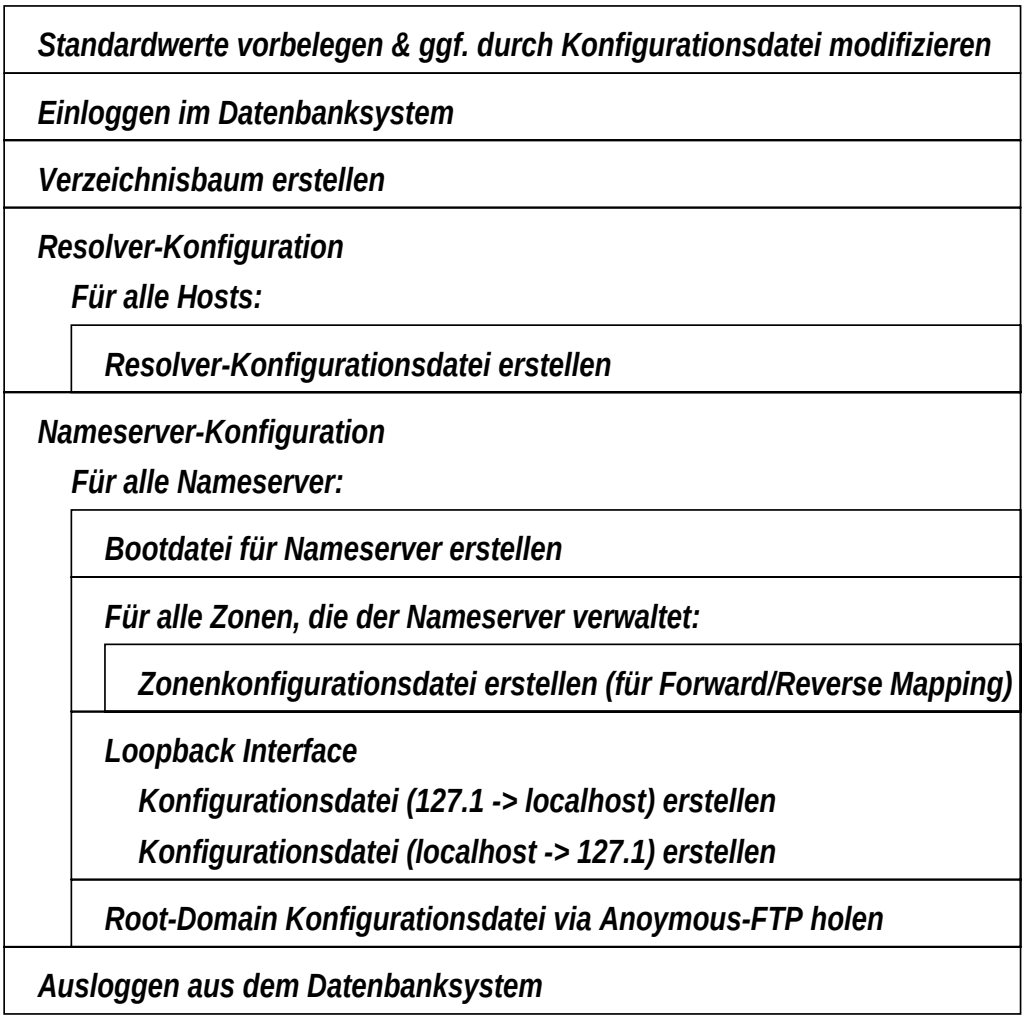


Abbildung 6.2: Programm *ora2dns* – Übersicht

Maßnahmen erheblich umfangreicher sind. Diese sollen d
gramm aufbereitete Parameter en
weis

der Namerserver läuft bestimmt werden. Sind einzelne Attribute dabei
tenbank nicht spezifiziert, so werden sie
(vgl. Kapit

Si

6.4.1 Verzeichnisbaum aufbauen

Das Modul *make_dir.pl* (Listing siehe Anhang A.4) beinhaltet im wesentl
eine Prozedur, mit deren Hilfe die Unterverze
Konfiguration

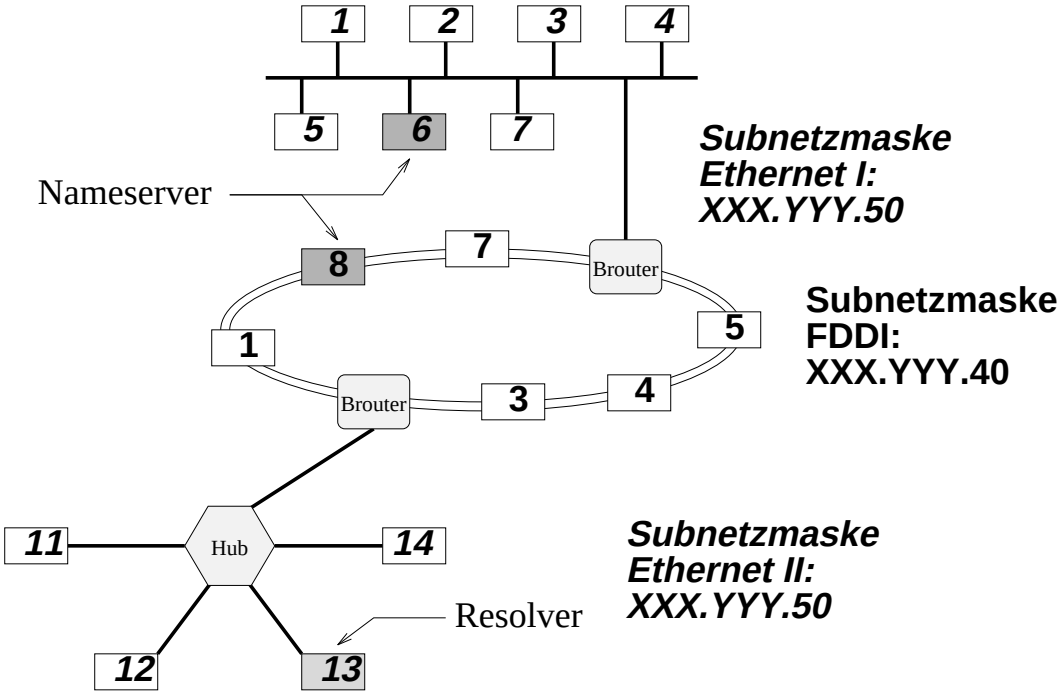


Abbildung 6.3: Geteiltes Subnetz

Service anbietet. Hierzu zählen auch die Standardeintr
terface. Als nächstes fol

Host- und ggf. Domainnamen bezeichnet. Dadurch können Glue Records
derlich werden. Alle Kandidaten hierfür wer

Bei den Secon

6.4.6 Loopback Interface (127.1 → localhost)

Das Modul *named_local.pl* (Listing siehe Anhang A.9). ist für den Reve
ping Teil des Loopback Interfaces zuständ

Neben dem Kopf der Konfigu
el

Wird das Programmdamit zumLaufen gebacht, so erscheint folgende Ausgab

```
sun1.(0)./home/a/a2824ch/dipl/perl> ora2dns
Starting execution...
Creating directory '/home/a/a2824ch/dipl/perl/ftp/dfvgate.lrz-muenchen.de'
Creating directory '/home/a/a2824ch/dipl/perl/ftp/dfvgate.lrz-muenchen.de/e
[...]
Creating directory '/home/a/a2824ch/dipl/perl/ftp/a2824ch.ppp.lrz-muenchen.
Creating directory '/home/a/a2824ch/dipl/perl/ftp/a2824ch.ppp.lrz-muenchen.
[...]
Creating directory '/home/a/a2824ch/dipl/perl/ftp/hp11.lrz-muenchen.de'
Creating directory '/home/a/a2824ch/dipl/perl/ftp/hp11.lrz-muenchen.de/etc'
Creating directory '/home/a/a2824ch/dipl/perl/ftp/dfvgate.lrz-muenchen.de/e
(remote-file) (local-file)
Creating directory '/home/a/a2824ch/dipl/perl/ftp/hp10.lrz-muenchen.de/etc/
(remote-file) (local-file)
Ready.
```

Nach einer Startmeldung, die erscheint, wenn der
text fertig vor übe

Die nun folgende Aufgabe seitens des DNS-Administrators besteht da
Dateien auf die einzelnen Hosts zu verte
Resolver) zu

Kapitel 7

Ausblick

In der vorliegenden Arbeit wurden Möglichkeiten
Name Systems aufgezeigt

Das in dieser Arbeit entwickelte Werkzeug wurde nicht für den isol
satz konzipiert, sondern für die Einb
mögliche

Daneben ließe sich aber auch ein eigenständiges Schema entwickeln, das
besondere, für das Routing wichtige Aspekte wi
mit dazwischenliegen

eine untergeordnete Rolle. Deshalb wird dafür seitens des ir
ten Verfahrens auch nur ein auf das
geboten (s

Um sich beim Einsatz des Verfahrens eine erhebliche Menge Arbeit zu spa-
ren, die bei der Eingabe der Daten in die Datenb
Teilverfahren zu e

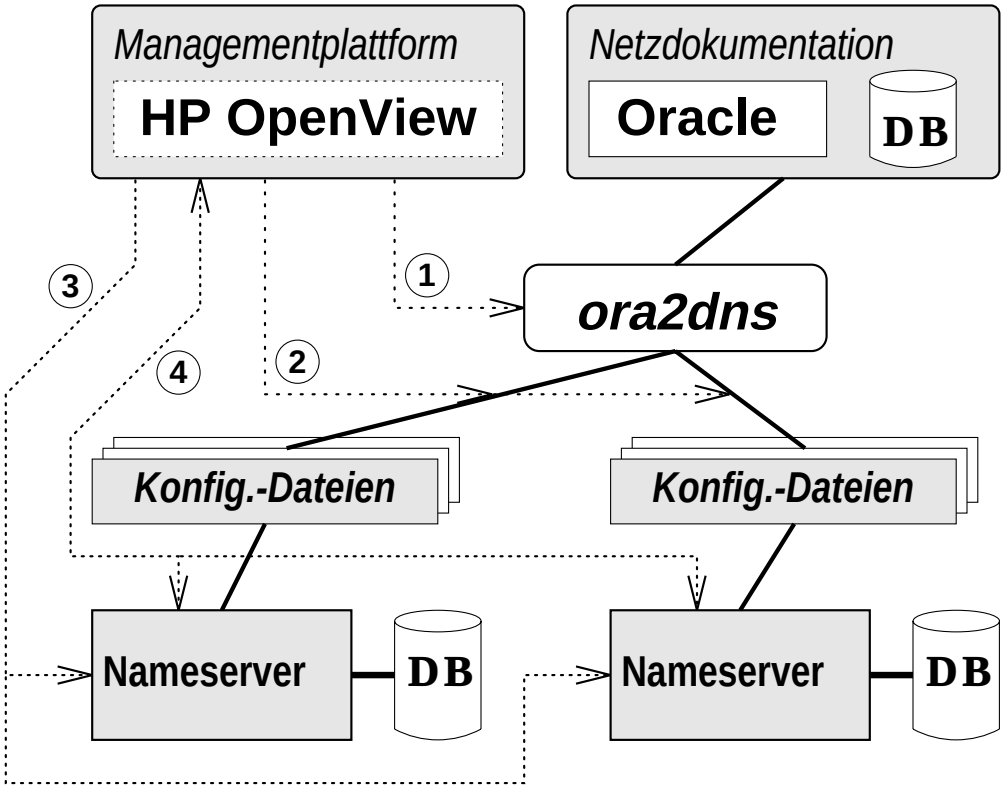


Abbildung 7.1: Einbindung in eine Managementplattorm

Die direkte Anbindung der Managementplattforma
dieser beiden an ein Tro

Domainsicht: Bei der obengenannten Sicht ist es erforderlich, einen Na
verhost für eine bestimmte Zone zu kenne
formationen zugel

Anhang A

Listings

A.1 Konfigurationsdatei .ora2dnsrc

```
# ********************************************************************
# * Integration des Managements des Domain-Name-Systems in die       *
# * vorhandene Managementumgebung des Leibniz-Rechenzentrums (LRZ)   *
# * (Diplomarbeit an der TU-Muenchen, SS 94 - WS 94/95)              *
# * ---------------------------------------------------------------- *
# * Autor: Heiko Hauck (hauck@informatik.tu-muenchen.de)             *
# * Datei: .ora2dnsrc (Konfigurationsdatei fuer ora2dns)             *
# ********************************************************************

# Format dieser Datei:
# <Variablenname> [<Leerzeichen> <Variablenbelegung>]
# Wird keine Variablenbelegung angegeben, so wird ein Leerstring verwendet.
# Kommentare muessen mit einem '#'-Zeichen beginnen.
# Leerzeilen sind erlaubt.

# DB-Qualifier, der bei saemtlichen DB-Anfragen vor den Tabellennamen
# eingefuegt wird
ORA2DNS.QUALIFIER
# Verzeichnis, in dem die erstellten Konfigurationsdateien
# (in entsprechenden Unterverzeichnissen) abgelegt werden
ORA2DNS.FTP_DIR /home/a/a2824ch/dipl/perl/ftp

# TTL-Wert fuer A-RRs in den Zonendateien
HOSTS.RECORD_TTL
# Klasse der A-RRs in den Zonendateien
HOSTS.RECORD_CLASS IN

# TTL-Wert fuer MX-RRs in den Zonendateien
MAILX.RECORD_TTL
# Klasse der MX-RRs in den Zonendateien
MAILX.RECORD_CLASS IN
```

```
# Dateiname fuer den Nameserverprozess
NAMESERVERS.FILENAME_EXE /etc/named
# Name der Datei, in der die ID des Nameserverprozesses steht
NAMESERVERS.FILENAME_PID /etc/named.pid
# Klasse des SOA-Records (fuer Loopback-Interface des Nameservers)
NAMESERVERS.SOA_CLASS IN
# Refresh-Intervall des SOA-Records (fuer Loopback-Interface des Nameservers)
NAMESERVERS.SOA_REFRESH 21600
# Retry-Intervall des SOA-Records (fuer Loopback-Interface des Nameservers)
NAMESERVERS.SOA_RETRY 3600
# Expire-Intervall des SOA-Records (fuer Loopback-Interface des Nameservers)
NAMESERVERS.SOA_EXPIRE 604800
# Minimum-Intervall des SOA-Records (fuer Loopback-Interface des Nameservers)
NAMESERVERS.SOA_MINIMUM 172800
# Verzeichnis, in dem die Zonendateien abgelegt werden
NAMESERVERS.FILES_DIR /etc/namedb
# Name der Zonendatei fuer das Loopback-Interface (127.1 ->localhost)
NAMESERVERS.FILENAME_LO named.local
# Name der Zonendatei fuer das Loopback-Interface (localhost -> 127.1)
NAMESERVERS.FILENAME_LB named.loopback
# Name der Root-Zonendatei (Cache Service)
NAMESERVERS.FILENAME_CA named.cache
# Host, von dem diese Datei mittels Anonymous FTP geholt wird
NAMESERVERS.FTP_HOST rs.internic.net
# Verzeichnis auf diesem Host
NAMESERVERS.FTP_DIR /domain
# Dateiname auf diesem Host
NAMESERVERS.FTP_FILE named.cache

# Dateiendung, die an den Zonendateinamen angefuegt wird, falls hierfuer
# der jeweilige Domainname verwendet wird (lokale Zonen - Secondary Service)
SECONDARY_LL.BACKUPFILE_SUFFIX .BF

# Dateiendung, die an den Zonendateinamen angefuegt wird, falls hierfuer
# der jeweilige Domainname verwendet wird (fremde Zonen - Secondary Service)
SECONDARY_LF.BACKUPFILE_SUFFIX .BF

# Dateiendung, die an den Zonendateinamen angefuegt wird, falls hierfuer
# der jeweilige Domainname verwendet wird
ZONES.SOURCEFILE_SUFFIX .DNF
# Klasse des SOA-Records
ZONES.SOA_CLASS IN
# Refresh-Intervall des SOA-Records
ZONES.SOA_REFRESH 21600
# Retry-Intervall des SOA-Records
ZONES.SOA_RETRY 3600
# Expire-Intervall des SOA-Records
ZONES.SOA_EXPIRE 604800
# Minimum-Intervall des SOA-Records
ZONES.SOA_MINIMUM 172800
```

A.2 Hauptprogramm ora2dns

```
#!/home/a/a2824ch/bin/oraperl
# ************************************************************************
# * Integration des Managements des Domain-Name-Systems in die           *
# * vorhandene Managementumgebung des Leibniz-Rechenzentrums (LRZ)       *
# * (Diplomarbeit an der TU-Muenchen, SS 94 - WS 94/95)                  *
# * -------------------------------------------------------------------- *
# * Autor: Heiko Hauck (hauck@informatik.tu-muenchen.de)                 *
# * Datei: ora2dns (Hauptprogramm)                                       *
# ************************************************************************

# AUFRUF: ora2dns [username password]
# Werden Benutzername und Passwort nicht mit angegeben,
# so werden hierfuer Standardangaben eingesetzt
# (siehe Prozedur login in dem Modul 'ora2dns.pl')

# Patch fuer die lokal installierte Version von 'oraperl'
@INC = ('.', '/sw/PD/perl-4.36/lib');

# Einbinden der Programmodule
require 'ora2dns.pl';
require 'make_dir.pl';
require 'res_conf.pl';
require 'named_boot.pl';
require 'named_cache.pl';
require 'named_forward.pl';
require 'named_reverse.pl';
require 'named_local.pl';
require 'named_loopback.pl';

# Programmstart
print "Starting execution...\n";

# Setzen bzw. Einlesen der Defaultwerte (aus Datei '$HOME/.ora2dnsrc')
&set_default;

# DB-Qualifier (globale Variable)
# wird bei den SQL-Abfragen vor alle Tabellennamen gesetzt
$DB = $def{'ORA2DNS.QUALIFIER'};

# Einloggen in die Datenbank
&login;

# Generierung der Host-Unterverzeichnisse
&make_dir;

# Resolverkonfiguration fuer jeden Host
&res_conf;

# ========================================================================
```

```
# Nameserverkonfiguration

# hierzu notwendige (globale) SQL-Anfragen:

# Auswahl saemtlicher Nameserver mit ihren Daten
&query(<<"END_SQL", *nameservers);
  select id, host, soa_responsible, soa_class, soa_serial, soa_refresh,
         soa_retry, soa_expire, soa_minimum, files_dir, filename_lo,
         filename_lb, filename_ca, ftp_host, ftp_dir, ftp_file
  from ${DB}nameservers
END_SQL

# Auswahl der Daten eines bestimmten Hosts und der zughoerigen Zone
$csr_hosts = &query_open(<<"END_SQL");
  select h.name, h.subdomain_name, h.zone, h.record_ttl, h.record_class,
         z.name, z.soa_responsible, z.soa_class, z.soa_serial,
         z.soa_refresh, z.soa_retry, z.soa_expire, z.soa_minimum
  from ${DB}hosts h, ${DB}zones z
  where h.zone = z.id and h.id = :1
END_SQL

# Auswahl aller IP-Adressen eines bestimmten Hosts
$csr_ipaddr = &query_open(<<"END_SQL");
  select ip_addr from ${DB}ipaddr where host = :1
END_SQL

# Auswahl aller Zonen, die einen bestimmten Primary Nameserver haben
$csr_zones = &query_open(<<"END_SQL");
  select id, name, sourcefile, soa_responsible, soa_class, soa_serial,
         soa_refresh, soa_retry, soa_expire, soa_minimum
  from ${DB}zones
  where prim_nameserver = :1
END_SQL

# ------------------------------------------------------------------------

# Schleife ueber saemtliche Nameserver

for(@nameservers)
{
  # Einlesen der Nameserverdaten

  local($id, $host, $soa_responsible, $soa_class, $soa_serial, $soa_refresh,
   $soa_retry, $soa_expire, $soa_minimum, $files_dir, $filename_lo,
   $filename_lb, $filename_ca, $ftp_host, $ftp_dir, $ftp_file) = split(/$;/);
  # $id, $host koennen nicht NULL sein
  # die restlichen Variablen duerfen NULL sein

  # Einlesen der Daten des zugehorigen Hosts

  local(@hosts);
```

```
&query_bind($csr_hosts, *hosts, $host);
# Ergebnis kann nur aus einem Tupel bestehen, da der Schluessel zones.id
# eindeutig ist (unique index)
local($hostname, $subdomainname, $zone, $record_ttl, $record_class,
 $domainname, $z_soa_responsible, $z_soa_class, $z_soa_serial,
 $z_soa_refresh, $z_soa_retry, $z_soa_expire, $z_soa_minimum)
 = split(/$;/, $hosts[0]);
# $hostname, $zone, $domainname, $z_soa_responsible & $z_soa_serial koennen
# nicht NULL sein
# $subdomainname, $record_ttl, $record_class, $z_soa_class, $z_soa_refresh,
# $z_soa_retry, $z_soa_expire & $z_soa_minimum duerfen NULL sein

# ---------------------------------------------------------------------

# Belegen der optionalen Attribute, falls
# sie in der Datenbank nicht belegt wurden (NULL-Werte)

# Eintraege fuer die SOA-Records der Zonen des Loopback-Interfaces
# (Forward & Reverse Mapping)

# Sind die folgenden beiden Attribute nicht speziell fuer den
# Nameserver angegeben, so werden die Werte der Zone, zu der
# der Nameserver-Host gehoert, verwendet

if(&testnull($soa_responsible))
  { $soa_responsible = $z_soa_responsible; }
if(&testnull($soa_serial))
  { $soa_serial = $z_soa_serial; }

# Im Gegensatz zu den beiden obigen Attributen kommt bei den folgenden
# noch der Default-Wert als letzte Alternative hinzu

if(&testnull($soa_class))
  { $soa_class = &testnull($z_soa_class) ?
                 $def{'NAMESERVERS.SOA_CLASS'} : $z_soa_class; }
if(&testnull($soa_refresh))
  { $soa_refresh = &testnull($z_soa_refresh) ?
                   $def{'NAMESERVERS.SOA_REFRESH'} : $z_soa_refresh; }
if(&testnull($soa_retry))
  { $soa_retry = &testnull($z_soa_retry) ?
                 $def{'NAMESERVERS.SOA_RETRY'} : $z_soa_retry; }
if(&testnull($soa_expire))
  { $soa_expire = &testnull($z_soa_expire) ?
                  $def{'NAMESERVERS.SOA_EXPIRE'} : $z_soa_expire; }
if(&testnull($soa_minimum))
  { $soa_minimum = &testnull($z_soa_minimum) ?
                   $def{'NAMESERVERS.SOA_MINIMUM'} : $z_soa_minimum; }

# Bei den folgenden Attributen stehen neben Nameserver-spezifischen
# Angaben Default-Werte zur Verfuegung
```

```
# Verzeichnis, in dem die Zonendateien abgelegt werden
if(&testnull($files_dir))
  { $files_dir = $def{'NAMESERVERS.FILES_DIR'}; }

# Name fuer Zonendatei des Loopback-Interfaces (127.1 -> localhost)
if(&testnull($filename_lo))
  { $filename_lo = $def{'NAMESERVERS.FILENAME_LO'}; }

# Name fuer Zonendatei des Loopback-Interfaces (localhost -> 127.1)
if(&testnull($filename_lb))
  { $filename_lb = $def{'NAMESERVERS.FILENAME_LB'}; }

# Name fuer Zonendatei fuer Root-Domain (Cache)
if(&testnull($filename_ca))
  { $filename_ca = $def{'NAMESERVERS.FILENAME_CA'}; }

# FTP Host, von dem obige Datei geholt wird
if(&testnull($ftp_host))
  { $ftp_host = $def{'NAMESERVERS.FTP_HOST'}; }

# Quellverzeichnis auf dem FTP-Server
if(&testnull($ftp_dir))
  { $ftp_dir = $def{'NAMESERVERS.FTP_DIR'}; }

# Name der Datei auf dem FTP-Server
if(&testnull($ftp_file))
  { $ftp_file = $def{'NAMESERVERS.FTP_FILE'}; }

# Die folgenden beiden Attribute werden u.U. fuer das Erzeugen
# eines Glue-Records fuer den aktuellen Nameserver-Host beim
# Forward-Mapping benoetigt; neben Host-spezifischen Angaben
# stehen Default-Werte zur Verfuegung

if(&testnull($record_ttl))
  { $record_ttl = $def{'HOSTS.RECORD_TTL'}; }
if(&testnull($record_class))
  { $record_class = $def{'HOSTS.RECORD_CLASS'}; }

# ------------------------------------------------------------------

# Aufbau des Fully Qualified Domain Name fuer den Nameserver-Host

local($fqdn);
if((!&testnull($subdomainname)) && (length($subdomainname) > 0))
{
  $fqdn = $hostname . '.' . $subdomainname . '.' . $domainname;
}
else
{
  $fqdn = $hostname . '.' . $domainname;
}
```

```
# Generierung der zugehoerigen IP-Adresse
# Im Falle mehrerer IP-Adressen wird die "erste" ausgewaehlt
# Die Liste der IP-Adressen wird ausserdem fuer die Boot-Datei benoetigt

local(@ipaddr);
&query_bind($csr_ipaddr, *ipaddr, $host);
# $ipaddr[..] kann nicht NULL sein
local($ipa) = $ipaddr[0];

# Loeschen von Leerzeichen in der Adresse (DB-Konvention)
$ipa =~ s/\x20//g;

# ---------------------------------------------------------------------

# Verzeichnis, in dem Daten fuer den Nameserver-Host abgelegt werden

local($directory) = $def{'ORA2DNS.FTP_DIR'} . '/' . $fqdn;

# Loeschen fuehrender Slashes ('/') in den Namesattributen

$files_dir =~ s/^\///o;
$filename_lo =~ s/^\///o;
$filename_lb =~ s/^\///o;
$filename_ca =~ s/^\///o;

# Generierung des Verzeichnisses fuer die Zonendateien

&make_dir_ns($directory, $files_dir);

# ---------------------------------------------------------------------

# Generierung der Boot-Datei

# Die Standard-Ausgabe wird fuer das Unterprogramm auf die
# entprechende Datei umgelenkt

local($filename) = $directory . '/etc/named.boot';
open(NF, ">$filename") || die "Can't open $filename: $!\n";
local($OLD_HANDLE) = select(NF);
&named_boot($ipa, $fqdn, $id, $filename_ca, *ipaddr, $files_dir,
            $filename_lo, $filename_lb);
select($OLD_HANDLE);
close(NF);

# ---------------------------------------------------------------------

# Verzeichnis fuer Zonendateien

$directory .= '/' . $files_dir. '/';
```

```
# ------------------------------------------------------------------

# Generierung der Zonen-Dateien fuer Forward- & Reverse-Mapping

local(@zones);
&query_bind($csr_zones, *zones, $id);

# Schleife ueber saemtliche Zonen, die von dem aktuellen Nameserver
# als Primary Nameserver verwaltet werden

for(@zones)
{
  local($id, $name, $sourcefile, $soa_responsible, $soa_class,
   $soa_serial, $soa_refresh, $soa_retry, $soa_expire, $soa_minimum)
   = split(/$;/);
  # $id, $name, $soa_responsible & $soa_serial koennen nicht NULL sein
  # $sourcefile, $soa_class, $soa_refresh, $soa_retry, $soa_expire &
  # $soa_minimum duerfen NULL sein

  # Belegen der optionalen Attribute, falls
  # sie in der Datenbank nicht belegt wurden (NULL-Werte)

  # Sind die folgenden Attribute nicht speziell fuer Zone angegeben,
  # so werden Default-Werte verwendet

  # Endung des Namens der Zonendatei

  if(&testnull($sourcefile))
  {
    $sourcefile = $name . $def{'ZONES.SOURCEFILE_SUFFIX'};
  }

  # Eintraege fuer den SOA-Record der Zone

  if(&testnull($soa_class))
    { $soa_class = $def{'ZONES.SOA_CLASS'}; }
  if(&testnull($soa_refresh))
    { $soa_refresh = $def{'ZONES.SOA_REFRESH'}; }
  if(&testnull($soa_retry))
    { $soa_retry = $def{'ZONES.SOA_RETRY'}; }
  if(&testnull($soa_expire))
    { $soa_expire = $def{'ZONES.SOA_EXPIRE'}; }
  if(&testnull($soa_minimum))
    { $soa_minimum = $def{'ZONES.SOA_MINIMUM'}; }

  # ------------------------------------------------------------------

  # Die Standard-Ausgabe wird fuer das jeweilige Unterprogramm auf die
  # entprechende Datei umgelenkt

  $filename = $directory . $sourcefile;
```

```
  open(NF, ">$filename") || die "Can't open $filename: $!\n";
  $OLD_HANDLE = select(NF);

  # ------------------------------------------------------------------

  # Forward- & Reverse-Mapping Zonendateien erfordern sehr unterschiedliche
  # Generierungsverfahren; Feststellung des Typs anhand des Zonennamens

  if(!($name =~ /\.in-addr\.arpa$/i))
  {
    &named_forward('/' . $files_dir. '/' . $sourcefile, $fqdn, $ipa,
             $soa_class, $soa_responsible, $soa_serial,
             $soa_refresh, $soa_retry, $soa_expire, $soa_minimum,
             $name, $id, $record_ttl, $record_class);
  }
  else
  {
    &named_reverse('/' . $files_dir. '/' . $sourcefile, $fqdn, $ipa,
             $soa_class, $soa_responsible, $soa_serial,
             $soa_refresh, $soa_retry, $soa_expire, $soa_minimum,
             $name, $id);
  }

  # ------------------------------------------------------------------

  select($OLD_HANDLE);
  close(NF);
}

# ----------------------------------------------------------------------

# Generierung der Zonen-Datei fuer das Loopback-Interface (127.1 -> localhost)

# Die Standard-Ausgabe wird fuer das Unterprogramm auf die
# entprechende Datei umgelenkt

$filename = $directory . $filename_lo;
open(NF, ">$filename") || die "Can't open $filename: $!\n";
$OLD_HANDLE = select(NF);
&named_local('/' . $files_dir. '/' . $filename_lo, $ipa,
             $soa_class, $fqdn, $soa_responsible, $soa_serial,
             $soa_refresh, $soa_retry, $soa_expire, $soa_minimum);
select($OLD_HANDLE);
close(NF);

# ----------------------------------------------------------------------

# Generierung der Zonen-Datei fuer das Loopback-Interface (localhost -> 127.1)

# Die Standard-Ausgabe wird fuer das Unterprogramm auf die
# entprechende Datei umgelenkt
```

```
  $filename = $directory . $filename_lb;
  open(NF, ">$filename") || die "Can't open $filename: $!\n";
  $OLD_HANDLE = select(NF);
  &named_loopback('/' . $files_dir. '/' . $filename_lb, $ipa,
                  $soa_class, $fqdn, $soa_responsible, $soa_serial,
                  $soa_refresh, $soa_retry, $soa_expire, $soa_minimum);
  select($OLD_HANDLE);
  close(NF);

  # ------------------------------------------------------------------

  # Holen der Zonen-Datei fuer die Root-Domain (via Anonymous FTP)
  # Als Passwort wird die E-Mail-Adresse des DNS-Administrators verwendet

  $filename = $directory . $filename_ca;
  &named_cache($filename, $ftp_host, $ftp_dir, $ftp_file, $soa_responsible);
}

# ------------------------------------------------------------------

# Schliessen der Cursor globaler SQL-Anfragen

&query_close($csr_hosts);
&query_close($csr_ipaddr);
&query_close($csr_zones);

# ==================================================================

# Ausloggen aus der Datenbank
&logout;

# Programmende
print "Ready.\n";
```

A.3 Modul ora2dns.pl

```
# ************************************************************************
# * Integration des Managements des Domain-Name-Systems in die         *
# * vorhandene Managementumgebung des Leibniz-Rechenzentrums (LRZ)     *
# * (Diplomarbeit an der TU-Muenchen, SS 94 - WS 94/95)                *
# * ---------------------------------------------------------------- *
# * Autor: Heiko Hauck (hauck@informatik.tu-muenchen.de)               *
# * Datei: ora2dns.pl (Oraperl Bibliothek fuer ora2dns)                *
# ************************************************************************

# *globale* Variable, auf deren Wert ein DB-Attribut gesetzt wird (vor
# dem Packen), wenn es undefiniert (NULL) ist

$NULL="\0";

# ************************************************************************

# PROZEDUR zum Einlesen der Standard-/Default-Werte (keine Parameter)
# Diese koennen mit Hilfe der Datei $HOME/.ora2dnsrc abgeaendert werden.
# Ergebnis: *globales* assoziatives Array %def

sub set_default
{
  # Standardvorbelegung des Arrays
  %def =
  (
    'ORA2DNS.QUALIFIER', '',
    'ORA2DNS.FTP_DIR', '/home/a/a2824ch/dipl/perl/ftp',
    'HOSTS.RECORD_TTL', '',
    'HOSTS.RECORD_CLASS', 'IN',
    'MAILX.RECORD_TTL', '',
    'MAILX.RECORD_CLASS', 'IN',
    'NAMESERVERS.FILENAME_EXE', '/etc/named',
    'NAMESERVERS.FILENAME_PID', '/etc/named.pid',
    'NAMESERVERS.SOA_CLASS', 'IN',
    'NAMESERVERS.SOA_REFRESH', '21600',
    'NAMESERVERS.SOA_RETRY', '3600',
    'NAMESERVERS.SOA_EXPIRE', '604800',
    'NAMESERVERS.SOA_MINIMUM', '172800',
    'NAMESERVERS.FILES_DIR', '/etc/namedb',
    'NAMESERVERS.FILENAME_LO', 'named.local',
    'NAMESERVERS.FILENAME_LB', 'named.loopback',
    'NAMESERVERS.FILENAME_CA', 'named.cache',
    'NAMESERVERS.FTP_HOST', 'rs.internic.net',
    'NAMESERVERS.FTP_DIR', '/domain',
    'NAMESERVERS.FTP_FILE', 'named.cache',
    'SECONDARY_LL.BACKUPFILE_SUFFIX', '.BF',
    'SECONDARY_LF.BACKUPFILE_SUFFIX', '.BF',
    'ZONES.SOURCEFILE_SUFFIX', '.DNF',
    'ZONES.SOA_CLASS', 'IN',
```

```
  'ZONES.SOA_REFRESH', '21600',
  'ZONES.SOA_RETRY', '3600',
  'ZONES.SOA_EXPIRE', '604800',
  'ZONES.SOA_MINIMUM', '172800'
);

# Name der Konfigurationsdatei
local($filename) = $ENV{'HOME'} . '/.ora2dnsrc';

# Aktuelle Zeilennummer beim Auslesen (fuer Fehlermeldungen)
local($line) = 1;

# Falls die Datei existiert, ...
if(-e $filename)
{
  # ..., wird sie geoeffnet
  open(RCFILE,$filename) || die "Can't open $filename: $!\n";

  # zeilenweises Einlesen
  while(<RCFILE>)
  {
    # Kommentarzeilen werden ignoriert
    if (/^#/) { $line++; next; }

    # Einlesen von Variablenname & -belegung
    local($var, $value) = ('', '');
    ($var, $value) = /^(\S+)\s+(\S+)/;

    # Ist die Belegung leer, so wird der Name dennoch eingelesen
    if($value eq '') { ($var) = /^(\S+)/; }

    # Variablenname gegeben?
    if($var ne '')
    {
      # Im Falle eines undefinierten Namens wird eine Fehlermeldung
      # ausgegeben (kein Programmabbruch)
      if(!defined($def{$var}))
      {
        print STDERR "Unknown variable $var in '$filename' line $line\n";
      }

      else
      # gueltiger Variablenname
      {
        # Folgende Variablen muessen mit einem Wert belegt sein,
        # sonst Fehlermeldung und keine Modifikation des bisherigen Wertes
        if((length($value) == 0) &&
           ($var ne 'ORA2DNS.QUALIFIER') &&
           ($var ne 'HOSTS.RECORD_TTL') &&
           ($var ne 'MAILX.RECORD_TTL') &&
           ($var ne 'SECONDARY_LL.BACKUPFILE_SUFFIX') &&
```

```
            ($var ne 'SECONDARY_LF.BACKUPFILE_SUFFIX') &&
            ($var ne 'ZONES.SOURCEFILE_SUFFIX'))
          {
            print STDERR
              "Variable $var in '$filename' line $line can't be set to ''\n";
          }

          else
          # Modifikation der Variable
          {
            $def{$var} = $value;
          }
        }
      }

      # neue Zeile
      $line++;
    }

    # Konfigurationsdatei schliessen
    close(RCFILE);
  }
}

# ***************************************************************************

# PROZEDUR zum Einloggen in die Datenbank
# (keine eigenen Parameter, sondern ggf. die des Programmaufrufes)
# Ergebnis: *globale* Variable $lda (Datenbank-ID)

sub login
{
  # lokale Variablen
  local($login_sid) = 'oracle';
  local($login_name,$login_pw);

  # Werden beim Programmstart zwei (oder mehrere) Parameter angegeben,
  # so werden diese (die ersten beiden) als Login-Name bzw. Passwort
  # fuer die Datenbank verwendet, ...
  if(@ARGV >= 2)
  {
    $login_name = @ARGV[0];
    $login_pw   = @ARGV[1];
  }

  else
  # ..., sonst werden Standardvorgaben eingesetzt:
  {
    # Die folgenden Zeilen sind an die jeweilige Umgebung anzupassen
    $login_name = `cat /home/a/a2824ch/oracle/LOGIN`;
    $login_pw   = `cat /home/a/a2824ch/oracle/PASSWD`;
```

```
#    $login_name = 'nduser';
#    $login_pw   = 'usernd';
  }

  # Einloggen
  # Misslingt dies, dann Programmabbruch
  $lda = &ora_login($login_sid, $login_name, $login_pw) ||
    die "$ora_errstr\n";
}

# ***************************************************************************

# PROZEDUR zum Ausloggen aus der Datenbank (keine Parameter)
# Voraussetzung: *globale* Variable $lda (Datenbank-ID) vorher gesetzt

sub logout
{
  # Ausloggen
  # Misslingt dies, dann Programmabbruch
  do ora_logoff($lda) || die "Can't log off Oracle\n";
}

# ***************************************************************************

# PROZEDUR fuer eine vollstaendige Datenbankabfrage
# Parameter: $_[0] = Query-Text
#            $_[1] = Pointer auf die Ergebnisliste mit Tupeln
# Im Falle mehrerer abgefragter Attribute werden diese gepackt.
# Voraussetzung: *globale* Variable $lda (Datenbank-ID) vorher gesetzt
# Rueckgabewert: Anzahl der gefundenen Tupel

sub query
{
  # (lokalen) Cursor oeffnen und Abfrage durchfuehren
  # Misslingt dies, dann Programmabbruch
  local($csr) = &ora_open($lda, $_[0]) || die $ora_errstr;

  # lokale Variablen
  local(@entry);
  local(*entrylist) = $_[1];
  local($entrycount) = 0;

  # Liste am Anfang leer
  @entrylist = ();

  # Einlesen saemtlicher Tupel
  while(@entry = &ora_fetch($csr))
  {
    # Ueberpruefung, ob <NULL>-Werte in den Attributen
    # vorhanden sind; in diesem Fall werden sie durch den Wert der
    # globalen Variable $NULL ersetzt (dieser darf also nicht als
```

```
    # Attributbelegung in der DB vorkommen).

    # Wuerde man auf diese Massnahme verzichten, so wuerde
    # die Information eines <NULL>-Wertes nach join/split
    # dahingehend verloren gehen, dass er als Leerstring
    # bzw. numerische Null interpretiert werden wuerde.

    # Ueberpruefung fuer jedes einzelne Attribut
    foreach(@entry)
    {
      $_ = $NULL unless defined($_);
    }

    # Packen aller Attribute eines Tupels zu einem Listenelement
    # und Eintrag in die Ergebnisliste
    push(@entrylist, join($;, @entry));
    $entrycount++;
  }

  # Warnung, falls ein Fehler aufgetreten ist
  warn $ora_errstr if $ora_errno;

  # Cursor schliessen
  # Misslingt dies, dann Programmabbruch
  do ora_close($csr) || die "Can't close Oracle cursor\n";

  # Anzahl der gefundenen Tupel
  $entrycount;
}

# ***************************************************************************

# PROZEDUR zum Eroeffnen einer Datenbankabfrage
# Parameter: $_[0] = Query-Text
# Voraussetzung: *globale* Variable $lda (Datenbank-ID) vorher gesetzt
# Rueckgabewert: Cursor fuer die Datenbankabfrage

sub query_open
{
  # Cursor oeffnen
  # Misslingt dies, dann Programmabbruch
  &ora_open($lda, $_[0]) || die $ora_errstr;
}

# ***************************************************************************

# PROZEDUR zum Durchfuehren einer Datenbankabfrage
# Parameter: $_[0]   = Cursor fuer die Datenbankabfrage
#            $_[1]   = Pointer auf die Ergebnisliste mit Tupeln
#            $_[2..] = Werte der zu bindenden Query-Variablen
# Im Falle mehrerer abgefragter Attribute werden diese gepackt.
```

```
# Rueckgabewert: Anzahl der gefundenen Tupel

sub query_bind
{
  # lokale Variablen
  local($csr) = shift @_;
  local(@entry);
  local(*entrylist) = shift @_;
  local($entrycount) = 0;

  # Liste am Anfang leer
  @entrylist = ();

  # Binden der Query-Variablen und Abfrage durchfuehren
  # Misslingt dies, dann Programmabbruch
  &ora_bind($csr,@_) || die $ora_errstr;

  # Einlesen saemtlicher Tupel
  while(@entry = &ora_fetch($csr))
  {
    # Ueberpruefung, ob <NULL>-Werte in den Attributen
    # vorhanden sind; in diesem Fall werden sie durch den Wert der
    # globalen Variable $NULL ersetzt (dieser darf also nicht als
    # Attributbelegung in der DB vorkommen).

    # Wuerde man auf diese Massnahme verzichten, so wuerde
    # die Information eines <NULL>-Wertes nach join/split
    # dahingehend verloren gehen, dass er als Leerstring
    # bzw. numerische Null interpretiert werden wuerde.

    # Ueberpruefung fuer jedes einzelne Attribut
    foreach(@entry)
    {
      $_ = $NULL unless defined($_);
    }

    # Packen aller Attribute eines Tupels zu einem Listenelement
    # und Eintrag in die Ergebnisliste
    push(@entrylist, join($;, @entry));
    $entrycount++;
  }

  # Warnung, falls ein Fehler aufgetreten ist
  warn $ora_errstr if $ora_errno;

  # Anzahl der gefundenen Tupel
  $entrycount;
}

# ********************************************************************
```

```
# PROZEDUR zum Schliessen einer Datenbankabfrage
# Parameter: $_[0] = Cursor fuer die Datenbankabfrage

sub query_close
{
  # Cursor schliessen
  # Misslingt dies, dann Programmabbruch
  do ora_close($_[0]) || die "Can't close Oracle cursor\n";
}

# ********************************************************************

# FUNKTION zum Testen, ob ein Attribut undefiniert ist
# Parameter: $_[0] = entpacktes Attribut
# Rueckgabewert: TRUE, falls undefiniert, sonst FALSE

sub testnull
{
  $_[0] eq $NULL;
}

# ********************************************************************

# PROZEDUR zum Sicherstellen, dass ein Attribut definiert ist
# (sonst Programmabbruch)
# Parameter: $_[0] = entpacktes Attribut
#            $_[1] = Tabellenname
#            $_[2] = Attributname
# Voraussetzung: *globale* Variable $DB (Datenbank-Qualifier) vorher gesetzt

sub notnull
{
  # Programmabbruch mit Fehlermeldung, falls undefiniert
  die "Illegal entry (NULL) for $_[2] in table ${DB}$_[1]\n"
  if $_[0] eq $NULL;
}

# ********************************************************************
1;
```

A.4 Modul make_dir.pl

```
# ************************************************************************
# * Integration des Managements des Domain-Name-Systems in die           *
# * vorhandene Managementumgebung des Leibniz-Rechenzentrums (LRZ)       *
# * (Diplomarbeit an der TU-Muenchen, SS 94 - WS 94/95)                  *
# * -------------------------------------------------------------------- *
# * Autor: Heiko Hauck (hauck@informatik.tu-muenchen.de)                 *
# * Datei: make_dir.pl (Oraperl Bibliothek fuer ora2dns)                 *
# ************************************************************************

# PROZEDUR zum Erstellen eines einzelnen Verzeichnisses innerhalb des
# Unix-Dateibaumes
# Parameter: $_[0] = Name des Verzeichnisses

sub mk_dir
{
  # Das Verzeichnis wird nur erstellt, wenn es noch nicht vorhanden ist
  if(!(-d $_[0]))
  {
    # Erstellung mit Leserechten fuer jedermann
    if(!mkdir($_[0], 0755))
    {
      # Misslingt dies, dann Programmabbruch
      die "Can't create directory '$_[0]'";
    }

    # Bestaetigung
    print "Creating directory '$_[0]'\n";
  }
}

# ************************************************************************

# PROZEDUR zum Erstellen des fuer die Resolverkonfiguration benoetigten
# Verzeichnisbaumes (keine Parameter)

sub make_dir
{
  # Wurzel des neuen Verzeichnisbaumes
  local($dir) = $def{'ORA2DNS.FTP_DIR'};

  # Falls diese nicht exisiert, dann Programmabbruch
  if(!(-d $dir))
  {
    die "Directory '$dir' (for ftp) does not exist";
  }

  # Auswahl aller Hosts mit Name, Subdomain- und Domainname
  # Hierfuer werden innerhalb des angegebenen Verzeichnisses Unterverzeichnisse
  # angelegt
```

```
  local(@hosts);
  &query(<<"  END_SQL", *hosts);
    select h.name, h.subdomain_name, z.name
    from ${DB}hosts h, ${DB}zones z
    where h.zone = z.id
  END_SQL

  # ------------------------------------------------------------------

  # Schleife ueber alle Hosts
  for(@hosts)
  {
    # Parameter fuer den aktuellen Host
    local($host, $subdomain, $domain) = split(/$;/);
    # $host, $domain koennen nicht NULL sein
    # $subdomain darf NULL sein

    # Aufbau des FQDN, der den Unterverzeichnisnamen bestimmt
    local($fqdn);
    if((!&testnull($subdomain)) && (length($subdomain) > 0))
    {
      $fqdn = $host . '.' . $subdomain . '.' . $domain;
    }
    else
    {
      $fqdn = $host . '.' . $domain;
    }

    # Generieren des Unterverzeichnisses und eines darin enthaltenen
    # '/etc'-Verzeichnisses
    &mk_dir($dir . '/' . $fqdn);
    &mk_dir($dir . '/' . $fqdn . '/etc');
  }
}

# ************************************************************************

# PROZEDUR zum Erstellen des fuer die Zonenkonfigurationsdateien benoetigten
# Verzeichnisses
# Parameter: $_[0] = Verzeichnis, in dem das neue erstellt werden soll
#            $_[1] = Name des zu erstellenden Verzeichnis(-pfades)

sub make_dir_ns
{
  # lokale Parameterkopien
  local($dir,$newdir) = @_;

  # Liste der einzelnen Knoten im Verzeichnispfad
  local(@fdirs) = split(/\//, $newdir);
```

```
  # sukzessiver Aufbau des Verzeichnispfades
  for(@fdirs)
  {
    $dir .= '/' . $_;
    &mk_dir($dir);
  }
}

# ****************************************************************************
1;
```

A.5 Modul res_conf.pl

```
# ************************************************************************
# * Integration des Managements des Domain-Name-Systems in die           *
# * vorhandene Managementumgebung des Leibniz-Rechenzentrums (LRZ)       *
# * (Diplomarbeit an der TU-Muenchen, SS 94 - WS 94/95)                  *
# * -------------------------------------------------------------------- *
# * Autor: Heiko Hauck (hauck@informatik.tu-muenchen.de)                 *
# * Datei: res_conf.pl (Oraperl Bibliothek fuer ora2dns)                 *
# ************************************************************************

# PROZEDUR zur formatierten Ausgabe eines Nameserver-Eintrags
# Parameter: $_[0] = IP-Adresse des Nameservers
#            $_[1] = FQDN des Nameservers
# Pro Konfigurationsdatei max. 3 Nameserver (*globale* Variable $nsc)

sub print_nameserver
{
  # Noch ein Eintrag zulaessig?
  if($nsc < 3)
  {
    # formatierte Ausgabe und Zaehlererhoehung
    print "nameserver @_[0]", " " x (20 - length(@_[0])), "# @_[1]\n";
    $nsc++;
  }
}

# ************************************************************************

# PROZEDUR zur Bestimmung des "optimalen" Weges zwischen zwei Hosts
# mit jeweils einer oder mehreren IP-Adressen; dabei die die guenstigste
# Adresse des Zielhosts ausgewaehlt und aus der Zieladressenliste gestrichen

# Das hierzu verwendete Verfahren basiert auf einem Vergleich der
# Subnetzmasken; diese Prozedur kann evtl. durch eine bessere ersetzt
# werden (siehe Ausarbeitung)

# Parameter: $_[0] = Pointer auf Liste mit IP-Adressen des Quellhosts
#            $_[1] = Pointer auf Liste mit IP-Adressen des Zielhosts
# Rueckgabewert: s. oben

sub best_route
{
  # Lokale Parameterkopien und Variablen
  local(*sourcelist) = $_[0];
  local(*destlist) = $_[1];
  local($fits) = -1;               # Anzahl der bisher max. identischen Bytes
  local($offset);                  # Nummer der zugehoerigen Ziel-IP-Adresse

  # Schleife ueber alle Quellhost-IP-Adressen
  SOURCE:
```

```
for(@sourcelist)
{
  # Extrahieren der Quellhost-IP-Adresse
  local($a,$b,$c,$d) =
  /\x20{0,2}(\d{1,3})\.\x20{0,2}(\d{1,3})\.\x20{0,2}(\d{1,3})\.\x20{0,2}(\d{1,3})/;

  # Nummer der jeweiligen Zielhost-IP-Adresse
  local($o)=0;

  # Schleife ueber alle Zielhost-IP-Adressen
  for(@destlist)
  {
    # Extrahieren der Quellhost-IP-Adresse
    # Programmabbruch im Falle eines syntaktischen Fehlers
    die "Illegal entry for ip_addr in table ${DB}ipaddr\n" unless
    /\x20{0,2}(\d{1,3})\.\x20{0,2}(\d{1,3})\.\x20{0,2}(\d{1,3})\.\x20{0,2}(\d{1,3})/;

    # Stufenweiser Byte-Vergleich zur Bestimmung des Uebereinstimmungsgrades
    # beider Adressen
    local($f);
    if($1 == $a)
    {
      if($2 == $b)
      {
        if($3 == $c)
        {
          if($4 == $d) { $f = 4; } # Identische Adressen
          else { $f = 3; }         # Bytes 1-3 gleich
        }
        else { $f = 2; }           # Bytes 1-2 gleich
      }
      else { $f = 1; }             # Byte 1 gleich
    }
    else { $f = 0; }               # Voellig verschiedene Adressen

    # Verbesserung gegenueber bisher?
    if($fits < $f)
    {
      # Neuer Uebereinstimmungsgrad und Nummer
      $fits = $f;
      $offset = $o;

      # Bei identischen Adressen keine Verbesserung mehr moeglich,
      # deshalb Schleifenabbruch
      last SOURCE if($fits == 4);
    }
    $o++;
  }
}

# Auswahl und Ausschneiden der besten IP-Adresse aus der Zielhostliste
```

```
  splice(@destlist,$offset,1);
}

# ********************************************************************

# PROZEDUR zur Erstellung saemtlicher Resolverkonfigurationsdateien
# Parameter: keine

sub res_conf
{
  # Auswahl aller Hosts mit ihren Identifikatoren, Namen, Subdomain-Namen &
  # Identifikatoren der zugehoerigen Zonen
  local(@hosts);
  &query(<<"  END_SQL", *hosts);
    select id, name, subdomain_name, zone from ${DB}hosts
  END_SQL

  # Auswahl des Domain-Namens & Host-Identifikators, auf dem der
  # zugehoerige Primary Nameserver laeuft, fuer eine bestimmte Zone
  local($csr_1) = &query_open(<<"  END_SQL");
    select z.name, n.host from ${DB}nameservers n, zones z
    where n.id = z.prim_nameserver and z.id = :1
  END_SQL

  # Auswahl aller IP-Adressen eines bestimmten Hosts
  local($csr_2) = &query_open(<<"  END_SQL");
    select ip_addr from ${DB}ipaddr where host = :1
  END_SQL

  # Auswahl der Host-Identifikatoren von lokalen Secondary Nameservern,
  # die eine bestimmte lokale Zone spiegeln
  local($csr_3) = &query_open(<<"  END_SQL");
    select n.host
    from ${DB}nameservers n, ${DB}secondary_ll s
    where n.id = s.nameserver and s.zone = :1
  END_SQL

  # Auswahl der Hosts mit ihren Namen, Subdomain- & Domainnamen,
  # die eine bestimmte IP-Adresse besitzen
  local($csr_4) = &query_open(<<"  END_SQL");
    select h.name, h.subdomain_name, z.name
    from ${DB}hosts h, ${DB}ipaddr i, ${DB}zones z
    where h.id = i.host and h.zone=z.id and i.ip_addr = :1
  END_SQL

  # ------------------------------------------------------------------

  # Schleife ueber saemtliche Hosts
  for(@hosts)
  {
    # Anzahl der Nameserver pro Konfig.Datei: *globale* Variable $nsc
```

```
$nsc = 0;

# Parameter fuer den aktuellen Host
local($id, $name, $subdomain, $zone) = split(/$;/);
# $id, $name koennen nicht NULL sein
# $zone darf nicht NULL sein (&notnull ...)
# $subdomain_name darf NULL sein
&notnull($zone, 'hosts', 'zone');

# ------------------------------------------------------------------

# Auswahl des Primary Nameservers der Zone des aktuellen Hosts
local(@dn_pnsh);
&query_bind($csr_1, *dn_pnsh, $zone);
# Ergebnis = 1 Tupel
local($domain, $pns_host) = split(/$;/,@dn_pnsh[0]);
# $domain, $pns_host koennen nicht NULL sein

# Vollstaendiger Domainname
if((!&testnull($subdomain)) && (length($subdomain) > 0))
{
  $domain = $subdomain . '.' . $domain;
}

# Ausgabedatei oeffnen und Standardausgabe dorthin umlenken
local($filename) = $def{'ORA2DNS.FTP_DIR'} . '/' .
  $name . '.' . $domain . '/etc/resolv.conf';
open(NF,">$filename") || die "Can't open $filename: $!\n";
local($OLD_HANDLE)=select(NF);

# IP-Adresse des aktuellen Hosts bestimmen
local(@ipa_host);
&query_bind($csr_2, *ipa_host, $id);
# $ipa_host[..] kann nicht NULL sein

# Auswahl der "ersten" IP-Adresse
local($ipa) = $ipa_host[0];

# Eliminieren von darin enthaltenen Leerzeichen
$ipa =~ s/\x20//g;

# ------------------------------------------------------------------

# Ausgabe des Dateikopfes
print "# /etc/resolv.conf for $name.$domain [$ipa]\n";
print "# created by ora2dns (08/94 by Heiko Hauck; Diplomarbeit TUM)\n\n";
print "# User: ", `whoami`;
print "# Date: ", `date`, "\n";;
print "domain     $domain\n\n";

# ------------------------------------------------------------------
```

```
# Laeuft der Primary Nameserver auf dem aktuellen Host, so wird dieser
# sofort eingetragen, ...
local(@ipa_nsh);
if($psn_host == $id)
{
  &print_nameserver('127.0.0.1', 'localhost');
}

else
# ..., sonst wird seine "beste" IP-Adresse in eine Liste (@ipa_nsh)
# eingetragen
{
  local(@ipa);
  &query_bind($csr_2, *ipa, $pns_host);
  # $ipa[..] kann nicht NULL sein
  @ipa_nsh  = (&best_route(*ipa_host, *ipa));
}

# ---------------------------------------------------------------------

# Auswahl der Secondary Nameserver der Zone des aktuellen Hosts
local(@sns_hosts);
&query_bind($csr_3, *sns_hosts, $zone);
# $sns_hosts[..] kann nicht NULL sein

# Schleife ueber alle Secondary Nameserver
for(@sns_hosts)
{
  # Laeuft der Nameserver auf dem aktuellen Host, so wird dieser
  # sofort eingetragen, ...
  if($_ == $id)
  {
    if ($nsc == 0)
    {
      &print_nameserver('127.0.0.1', 'localhost');
    }
  }

  else
  # ..., sonst wird seine "beste" IP-Adresse in eine Liste (@ipa_nsh)
  # eingetragen
  {
    local(@ipa);
    &query_bind($csr_2, *ipa, $_);
    # $ipa[..] kann nicht NULL sein
    push(@ipa_nsh,&best_route(*ipa_host, *ipa));
  }
}

# ---------------------------------------------------------------------
```

```
    # Eintragen der (restlichen) Nameserver in die Datei

    # Schleife ueber die jeweils "besten" IP-Adressen der Nameserverhosts
    while(@ipa_nsh)
    {
      # Auswahl und Ausaschneiden der "besten" Adresse der Liste
      local($ipa) = &best_route(*ipa_host,*ipa_nsh);

      # Auswahl des zugehoerigen Hosts (Name, Subdomain- & Domainname)
      local(@hn_sn_dn);
      if(&query_bind($csr_4, *hn_sn_dn, $ipa) !=1 )
      {
        # Programmabbruch, falls eine IP-Adresse mehreren Hosts zugeordnet
        # werden kann
        die "Multiple hosts for one IP address";
      }
      local($host, $subdomain, $domain) = split(/$;/,$hn_sn_dn[0]);
      # $host, $domain koennen nicht NULL sein
      # $subdomain darf NULL sein

      # Vollstaendiger Domainname
      if((!&testnull($subdomain)) && (length($subdomain) > 0))
      {
        $domain = $subdomain . '.' . $domain;
      }

      # Eliminieren von darin enthaltenen Leerzeichen
      $ipa =~ s/\x20//g;

      # Eintrag
      &print_nameserver($ipa, $host . '.' . $domain);
    }

    # Standardausgabe wieder herstellen und Datei schliessen
    select($OLD_HANDLE);
    close(NF);
  }

  # Schliessen der oben fuer die Datenbankabfrage geoeffneten Cursor
  &query_close($csr_1);
  &query_close($csr_2);
  &query_close($csr_3);
  &query_close($csr_4);
}

# ****************************************************************************
1;
```

A.6 Modul named_boot.pl

```
# ********************************************************************
# * Integration des Managements des Domain-Name-Systems in die       *
# * vorhandene Managementumgebung des Leibniz-Rechenzentrums (LRZ)   *
# * (Diplomarbeit an der TU-Muenchen, SS 94 - WS 94/95)              *
# * ---------------------------------------------------------------- *
# * Autor: Heiko Hauck (hauck@informatik.tu-muenchen.de)             *
# * Datei: named_boot.pl (Oraperl Bibliothek fuer ora2dns)           *
# ********************************************************************

# PROZEDUR zur Generierung einer Nameserver-Bootdatei
# Parameter: $_[0] = IP-Adresse des Nameserverhosts (ohne Leerzeichen)
#            $_[1] = FQDN des Nameserverhosts
#            $_[2] = ID des Nameservers
#            $_[3] = Name der Root-Domain-Datei (Cache Service)
#            $_[4] = Pointer auf Liste mit IP-Adressen des Nameserverhosts
#            $_[5] = Verzeichnis, in dem die Zonendateien abgelegt werden
#            $_[6] = Name der Localhost-Zonendatei (127.1 -> localhost)
#            $_[7] = Name der Loopback-Zonendatei (localhost -> 127.1)
# Die Ausgabe erfolgt zur Standard-Ausgabe.

sub named_boot
{
  # Auswahl der Domainnamen und Sourcefile-Namen,
  # deren Zonen einen best. Primary Nameserver haben
  local($csr_prim) = &query_open(<<"  END_SQL");
    select name, sourcefile from ${DB}zones where prim_nameserver = :1
  END_SQL

  # Auswahl des Backupfile-Namens, Domainnamens & lokalen Primary Nameservers
  # fuer Zonen, die einen bestimmten lokalen Secondary Nameserver haben
  local($csr_sec_ll) = &query_open(<<"  END_SQL");
    select s.backupfile, z.name, z.prim_nameserver
    from ${DB}secondary_ll s, ${DB}zones z
    where z.id = s.zone and s.nameserver = :1
  END_SQL

  # Auswahl der IP-Adressen eines bestimmten Primary Nameservers
  local($csr_ipaddr) = &query_open(<<"  END_SQL");
    select i.ip_addr
    from ${DB}ipaddr i, ${DB}nameservers n
    where i.host = n.host and n.id = :1
  END_SQL

  # Auswahl fremder Domainnamen, deren Prim.Nameserver-IP-Adressen und
  # Backupfile-Namen, die einen bestimmten lokalen Secondary Nameserver haben
  local($csr_sec_lf) = &query_open(<<"  END_SQL");
    select zone_name, ip_addr, backupfile
    from ${DB}secondary_lf
    where nameserver = :1
```

```
END_SQL

# Auswahl der Forwarder-IP-Adressen eines bestimmten Slave Nameservers
local($csr_forward) = &query_open(<<"  END_SQL");
  select ip_addr from ${DB}forwarders where nameserver = :1
END_SQL

# Test, ob ein bestimmter Nameserver Slave Nameserver ist
local($csr_slave) = &query_open(<<"  END_SQL");
  select count(*) from ${DB}nameservers_slave where nameserver = :1
END_SQL

# ------------------------------------------------------------------

# Ausgabe des Dateikopfes
print "; /etc/named.boot for $_[1] [$_[0]]\n";
print "; created by ora2dns (08/94 by Heiko Hauck; Diplomarbeit TUM)\n";
print ";\n";
print "; User: ", `whoami`;
print "; Date: ", `date`;
print ";\n";
print "; type\t\tdomain\t\t\tsource file or host\tbackup file\n";
print ";\n";

# Ausgabe des Verzeichnisses, in dem die Zonendateien abgelegt werden
print "directory\t/$_[5]\n";

# ------------------------------------------------------------------

# Zonen-Eintraege fuer den Primary Service des Nameservers

# Hinweis als Kommentarzeile
print "; primary nameserver:\n";

# Endung der Zonendateinamen
local($suffix) = $def{'ZONES.SOURCEFILE_SUFFIX'};

# Zonen bestimmen
local(@prim);
&query_bind($csr_prim, *prim, $_[2]);

# Schleife ueber alle betroffenen Zonen
for(@prim)
{
  # Parameter der aktuellen Zone
  local($domainname, $sourcefile) = split(/$;/);
  # $domainname kann nicht NULL sein
  # $sourcefile darf NULL sein

  # Wird der Name der Zonendatei nicht explizit angegeben, ...
  if(&testnull($sourcefile))
```

```
  {
    # ..., so wird der Domainname + Suffix verwendet
    $sourcefile = $domainname . $suffix;
  }

  # Eintrag in die Bootdatei
  print "primary\t\t$domainname\t$sourcefile\n";
}

# Standardeintraege in der Bootdatei, da jeder Nameserver Primary Nameserver
# fuer das Loopback-Interface (forward & reverse) ist
print "primary\t\t0.0.127.in-addr.arpa\t$_[6]\n";
print "primary\t\tlocalhost\t\t$_[7]\n";

# ------------------------------------------------------------------

# Zonen-Eintraege fuer den Secondary Service des Nameservers

# Hinweis als Kommentarzeile
print "; secondary nameserver:\n";

# Endung der Zonendateinamen (lokale Zonen)
$suffix = $def{'SECONDARY_LL.BACKUPFILE_SUFFIX'};

# Lokale Zonen bestimmen
local(@sec_ll);
&query_bind($csr_sec_ll, *sec_ll, $_[2]);

# Schleife ueber alle betroffenen Zonen
for(@sec_ll)
{
  # Parameter der aktuellen Zone
  local($backupfile, $domainname, $prim_id) = split(/$;/);
  # $backupfile darf NULL sein
  # $domainname, $prim_id koennen nicht NULL sein

  # IP-Adressen des zugehoerigen Primary Nameservers bestimmen ...
  local(@ipaddr);
  &query_bind($csr_ipaddr, *ipaddr, $prim_id);
  # $ipaddr[..] kann nicht NULL sein

  # ... und die "beste" davon auswaehlen
  local($ipa) = &best_route($_[4], *ipaddr);

  # Eliminieren von darin enthaltenen Leerzeichen
  $ipa =~ s/\x20//g;

  # Wird der Name der Zonendatei nicht explizit angegeben, ...
  if(&testnull($backupfile))
  {
    # ..., so wird der Domainname + Suffix verwendet
```

```
    $backupfile = $domainname . $suffix;
  }

  # Eintrag in die Bootdatei
  print "secondary\t$domainname\t$ipa\t$backupfile\n";
}

# ------------------------------------------------------------------

# Endung der Zonendateinamen (fremde Zonen)
$suffix = $def{'SECONDARY_LF.BACKUPFILE_SUFFIX'};

# Fremde Zonen bestimmen
local(@sec_lf);
&query_bind($csr_sec_lf, *sec_lf, $_[2]);

# Schleife ueber alle betroffenen Zonen
for(@sec_lf)
{
  # Parameter der aktuellen Zone
  local($domainname, $ipa, $backupfile) = split(/$;/);
  # $domainname, $ipa koennen nicht NULL sein
  # $backupfile darf NULL sein

  # Eliminieren von in der IP-Adresse enthaltenen Leerzeichen
  $ipa =~ s/\x20//g;

  # Wird der Name der Zonendatei nicht explizit angegeben, ...
  if(&testnull($backupfile))
  {
    # ..., so wird der Domainname + Suffix verwendet
    $backupfile = $domainname . $suffix;
  }

  # Eintrag in die Bootdatei
  print "secondary\t$domainname\t$ipa\t$backupfile\n";
}

# ------------------------------------------------------------------

# Forwarding-Eintraege, Nameserver als Forwarder Nameserver arbeitet

# Forwarder-IP-Adressen bestimmen
local(@forward);
&query_bind($csr_forward, *forward, $_[2]);

# Sind welche vorhanden?
if(@forward > 0)
{
  # Hinweis als Kommentarzeile
  print "; forwarding nameservers:\n";
```

```
    # Eliminieren von in den IP-Adressen enthaltenen Leerzeichen
    for(@forward)
    {
      s/\x20//g;
    }

    # Eintrag in die Bootdatei
    print "forwarders\t", join(' ', @forward), "\n";

    # Laeuft der Nameserver als Slave, ...
    local(@slave);
    &query_bind($csr_slave, *slave, $_[2]);
    if($slave[0] > 0)
    {
      # ..., so gibt es einen entsprechenden Eintrag in die Bootdatei
      print "slave\n";
    }
  }

  # ---------------------------------------------------------------------

  # Standardeintrag in der Bootdatei, da jeder Nameserver Cache Nameserver
  # fuer die Root-Domain ist
  print "; cache nameserver:\n";
  print "cache\t\t.\t\t\t$_[3]\n";

  # Schliessen der oben fuer die Datenbankabfrage geoeffneten Cursor
  &query_close($csr_prim);
  &query_close($csr_sec_ll);
  &query_close($csr_ipaddr);
  &query_close($csr_sec_lf);
  &query_close($csr_forward);
  &query_close($csr_slave);
}

# ***********************************************************************
1;
```

A.7 Modul named_forward.pl

```
# ************************************************************************
# * Integration des Managements des Domain-Name-Systems in die           *
# * vorhandene Managementumgebung des Leibniz-Rechenzentrums (LRZ)       *
# * (Diplomarbeit an der TU-Muenchen, SS 94 - WS 94/95)                  *
# * -------------------------------------------------------------------- *
# * Autor: Heiko Hauck (hauck@informatik.tu-muenchen.de)                 *
# * Datei: named_forward.pl (Oraperl Bibliothek fuer ora2dns)            *
# ************************************************************************

# PROZEDUR zur Generierung einer Zonen-Datei fuer das Forward Mapping
# Parameter: $_[0]  = Name der Datei
#            $_[1]  = FQDN des Nameserverhosts
#            $_[2]  = IP-Adresse des Nameserverhosts
#            $_[3]  = Klasse des SOA-Records
#            $_[4]  = E-Mail-Adresse (SOA-Responsible)
#            $_[5]  = Seriennummer im SOA-Record
#            $_[6]  = Refresh-Intervall im SOA-Record
#            $_[7]  = Retry-Intervall im SOA-Record
#            $_[8]  = Expire-Intervall im SOA-Record
#            $_[9]  = Minimum-Intervall im SOA-Record
#            $_[10] = Domainname der Zone
#            $_[11] = Identifikator der Zone
#            $_[12] = TTL fuer Glue-Records
#            $_[13] = Klasse der Glue-Records
# Die Ausgabe erfolgt zur Standard-Ausgabe.

sub named_forward
{
  # Ausgabe des Dateikopfes
  print "; $_[0] for $_[1] [$_[2]]\n";
  print "; created by ora2dns (08/94 by Heiko Hauck; Diplomarbeit TUM)\n";
  print ";\n";
  print "; User: ", `whoami`;
  print "; Date: ", `date`;
  print ";\n";

  # Ausgabe der betroffenen Zone
  print "\$ORIGIN\t\t$_[10].\n";

  # Ausgabe des SOA-Records fuer diese Zone
  print "\@\t\t$_[3]\tSOA\t$_[1]. $_[4]. (\n";
  print "\t\t\t\t\t$_[5]   ; serial\n";
  print "\t\t\t\t\t$_[6]   ; refresh\n";
  print "\t\t\t\t\t$_[7]   ; retry\n";
  print "\t\t\t\t\t$_[8]   ; expire\n";
  print "\t\t\t\t\t$_[9] ) ; minimum\n";

  # ---------------------------------------------------------------------
```

```
# Eintrag fuer den Primary Nameserver der Zone
print "\t\tIN\tNS\t$_[1].\n";

# ------------------------------------------------------------------

# Anlegen einer Liste fuer potentielle Glue-Records
# folgende Attribute werden dabei als Listenelement gepackt:
# FQDN, TTL, Class, IP-Adresse (Reihenfolge wie angegeben)

# Primary Nameserver Host ist Kandidat fuer Glue-Record
local(@glue) = (join($;, ($_[1], $_[12], $_[13], $_[2])));

# ------------------------------------------------------------------

# Lokale Secondary Nameserver fuer die Zone
local(@servers);
&query(<<"  END_SQL", *servers);
  select h.id, h.name, h.subdomain_name, h.record_ttl, h.record_class, z.name
  from ${DB}zones z, ${DB}hosts h, ${DB}nameservers n, ${DB}secondary_ll s
  where z.id = h.zone and h.id = n.host and
        n.id = s.nameserver and s.zone = $_[11]
END_SQL

# Auswahl aller IP-Adressen eines bestimmten Hosts
local($csr_ipaddr) = &query_open(<<"  END_SQL");
  select ip_addr from ${DB}ipaddr where host = :1
END_SQL

# Schleife ueber alle betroffenen Nameserver
for(@servers)
{
  # Parameter des aktuellen Nameservers
  local($id, $hostname, $subdomainname, $record_ttl, $record_class,
   $domainname) = split(/$;/);
  # $id, $hostname, $domainname koennen nicht NULL sein
  # $subdomainname, $record_ttl, $record_class duerfen NULL sein

  # Aufbau des Fully Qualified Domain Name fuer den Nameserver-Host
  local($fqdn);
  if((!&testnull($subdomainname)) && (length($subdomainname) > 0))
  {
    $fqdn = $hostname . '.' . $subdomainname . '.' . $domainname;
  }
  else
  {
    $fqdn = $hostname . '.' . $domainname;
  }

  # Eintrag fuer den Nameserver
  print "\t\tIN\tNS\t$fqdn.\n";
```

```
  # ------------------------------------------------------------------

  # Secondary Nameserver Host ist Kandidat fuer Glue-Record

  # Record-TTL und -Class
  if(&testnull($record_ttl))
    { $record_ttl = $def{'HOSTS.RECORD_TTL'}; }
  if(&testnull($record_class))
    { $record_class = $def{'HOSTS.RECORD_CLASS'}; }

  # IP-Adresse des Nameserver-Hosts
  local(@ipaddr);
  &query_bind($csr_ipaddr, *ipaddr, $id);
  # $ipaddr[..] kann nicht NULL sein
  local($ipa) = $ipaddr[0];

  # Loeschen von Leerzeichen in der Adresse (DB-Konvention)
  $ipa =~ s/\x20//g;

  # Eintrag in die Liste
  push(@glue, join($;, ($fqdn, $record_ttl, $record_class, $ipa)));
}

# ------------------------------------------------------------------

# Achtung: 'Perl' ueberschreibt aktuelle Parameter ohne entspr. Zuweisung!
# Deshalb: Kopie in lokale Variable
local($current) = $_[10];

# ------------------------------------------------------------------

# Fremde Secondary Nameserver fuer die Zone

# Auswahl der FQDNs und IP-Adressen der betroffenen Nameserver
&query(<<"  END_SQL", *servers);
  select ip_name, ip_addr from ${DB}secondary_fl where zone = $_[11]
END_SQL

# Schleife ueber alle betroffenen Nameserver
for(@servers)
{
  # Parameter des aktuellen Nameservers
  local($ip_name, $ip_addr) = split(/$;/);
  # $ip_name kann nicht NULL sein
  # $ip_addr darf u.U. NULL sein

  # Eintrag fuer den Nameserver
  print "\t\tIN\tNS\t$ip_name.\n";

  # ------------------------------------------------------------------
```

```
  # Secondary Nameserver Host ist Kandidat fuer Glue-Record,
  # falls sein FQDN in der zur aktuellen Zone gehoerigen Domain liegt
  if($ip_name =~ /\.$current$/i)
  {
    # In diesem Fall muss seine IP-Adresse angegeben werden,
    # sonst Programmabbruch
    &notnull($ip_addr, 'ip_addr', 'secondary_fl');

    # Loeschen von Leerzeichen in der Adresse (DB-Konvention)
    $ip_addr =~ s/\x20//g;

    # Eintrag in die Liste
    push(@glue, join($;, ($ip_name, '', 'IN', $ip_addr)));
  }
}

# ---------------------------------------------------------------------

# Nameserver Eintraege fuer delegierte (lokale) Subdomains

# Auswahl aller Zonen, die die aktuelle Zone als Vaterzone haben
local(@domains);
&query(<<"  END_SQL", *domains);
  select id, name, prim_nameserver from ${DB}zones where parent = $_[11]
END_SQL

# Auswahl von ID, Namen, Subdomainnamen, RR-Parameter und Domainnamen
# eines Hosts, auf dem ein bestimmter Primary Nameserver laeuft
local($csr_prim) = &query_open(<<"  END_SQL");
  select h.id, h.name, h.subdomain_name, h.record_ttl, h.record_class, z.name
  from ${DB}zones z, ${DB}hosts h, ${DB}nameservers n
  where z.id = h.zone and h.id = n.host and n.id = :1
END_SQL

# Auswahl von ID, Namen, Subdomainnamen, RR-Parameter und Domainnamen
# von Hosts, auf denen fuer eine bestimmte Zone Secondary Nameserver laufen
local($csr_serv) = &query_open(<<"  END_SQL");
  select h.id, h.name, h.subdomain_name, h.record_ttl, h.record_class, z.name
  from ${DB}zones z, ${DB}hosts h, ${DB}nameservers n, ${DB}secondary_ll s
  where z.id = h.zone and h.id = n.host and
        n.id = s.nameserver and s.zone = :1
END_SQL

# Auswahl von fremden Secondary Nameservern fuer eine bestimmte Zone
local($csr_secf) = &query_open (<<"  END_SQL");
  select ip_name, ip_addr from ${DB}secondary_fl where zone = :1
END_SQL

# ---------------------------------------------------------------------

# Schleife ueber alle delegierten lokalen Subdomains
```

```
for(@domains)
{
  # Parameter der akuellen Subdomain
  local($id, $name, $prim_nameserver) = split(/$;/);
  # $id, $name & $prim_nameserver koennen nicht NULL sein

  # Subdomain-Name (innerhalb) der Domain bestimmen
  local($subdomain) = $name;
  $subdomain =~ s/\.$current$//i;

  # Primary Nameserver fuer Subdomain
  local(@prim);
  &query_bind($csr_prim, *prim, $prim_nameserver);
  local($hostid, $hostname, $subdomainname, $record_ttl,
        $record_class, $domainname) = split(/$;/, $prim[0]);
  # $hostid, $hostname, $domainname koennen nicht NULL sein
  # $subdomainname, $record_ttl, $record_class duerfen NULL sein

  # Aufbau des Fully Qualified Domain Name fuer den Primary Nameserver
  local($fqdn);
  if((!&testnull($subdomainname)) && (length($subdomainname) > 0))
  {
    $fqdn = $hostname . '.' . $subdomainname . '.' . $domainname;
  }
  else
  {
    $fqdn = $hostname . '.' . $domainname;
  }

  # Ausgabe des Subdomain-Namens mit Primary Nameserver
  print "$subdomain\t\tIN\tNS\t$fqdn.\n";

  # ------------------------------------------------------------------

  # Primary Nameserver Host ist Kandidat fuer Glue-Record

  # Record-TTL und -Class
  if(&testnull($record_ttl))
    { $record_ttl = $def{'HOSTS.RECORD_TTL'}; }
  if(&testnull($record_class))
    { $record_class = $def{'HOSTS.RECORD_CLASS'}; }

  # IP-Adresse des Nameserver-Hosts
  local(@ipaddr);
  &query_bind($csr_ipaddr, *ipaddr, $hostid);
  # $ipaddr[..] kann nicht NULL sein
  local($ipa) = $ipaddr[0];

  # Loeschen von Leerzeichen in der Adresse (DB-Konvention)
  $ipa =~ s/\x20//g;
```

```
# Eintrag in die Liste
push(@glue, join($;, ($fqdn, $record_ttl, $record_class, $ipa)));

# ------------------------------------------------------------------

# Lokale Secondary Nameserver fuer die Subdomain
local(@serv);
&query_bind($csr_serv, *serv, $id);

# Schleife ueber alle lokalen Secondary Nameserver
for(@serv)
{
  # Host-Parameter des aktuellen Nameservers
  local($id, $hostname, $subdomainname, $record_ttl, $record_class,
   $domainname) = split(/$;/);
  # $id, $hostname, $domainname koennen nicht NULL sein
  # $subdomainname, $record_ttl, $record_class duerfen NULL sein

  # Aufbau des Fully Qualified Domain Name fuer den Secondary Nameserver
  local($fqdn);
  if((!&testnull($subdomainname)) && (length($subdomainname) > 0))
  {
    $fqdn = $hostname . '.' . $subdomainname . '.' . $domainname;
  }
  else
  {
    $fqdn = $hostname . '.' . $domainname;
  }

  # Ausgabe des Secondary Nameservers
  print "\t\tIN\tNS\t$fqdn.\n";

  # ------------------------------------------------------------------

  # Secondary Nameserver Host ist Kandidat fuer Glue-Record

  # Record-TTL und -Class
  if(&testnull($record_ttl))
    { $record_ttl = $def{'HOSTS.RECORD_TTL'}; }
  if(&testnull($record_class))
    { $record_class = $def{'HOSTS.RECORD_CLASS'}; }

  # IP-Adresse des Nameserver-Hosts
  local(@ipaddr);
  &query_bind($csr_ipaddr, *ipaddr, $id);
  # $ipaddr[..] kann nicht NULL sein
  local($ipa) = $ipaddr[0];

  # Loeschen von Leerzeichen in der Adresse (DB-Konvention)
  $ipa =~ s/\x20//g;
```

```
    # Eintrag in die Liste
    push(@glue, join($;, ($fqdn, $record_ttl, $record_class, $ipa)));
  }

  # ------------------------------------------------------------------

  # Fremde Secondary Nameserver fuer die Subdomain
  local(@secf);
  &query_bind($csr_secf, *secf, $id);

  # Schleife ueber alle fremden Secondary Nameserver
  for(@secf)
  {
    # Parameter des aktuellen Nameservers
    local($ip_name, $ip_addr) = split(/$;/);
    # $ip_name kann nicht NULL sein
    # $ip_addr darf u.U. NULL sein

    # Eintrag fuer den Nameserver
    print "\t\tIN\tNS\t$ip_name.\n";

    # ----------------------------------------------------------------

    # Secondary Nameserver Host ist Kandidat fuer Glue-Record,
    # falls sein FQDN in der zur aktuellen Zone gehoerigen Domain liegt
    if($ip_name =~ /\.$current$/i)
    {
      # In diesem Fall muss seine IP-Adresse angegeben werden,
      # sonst Programmabbruch
      &notnull($ip_addr, 'ip_addr', 'secondary_fl');

      # Loeschen von Leerzeichen in der Adresse (DB-Konvention)
      $ip_addr =~ s/\x20//g;

      # Eintrag in die Liste
      push(@glue, join($;, ($ip_name, '', 'IN', $ip_addr)));
    }
  }
}

# Schliessen der fuer delegierte (lokale) Subdomains geoeffneten DB-Cursor
&query_close($csr_prim);
&query_close($csr_serv);
&query_close($csr_secf);

# ------------------------------------------------------------------

# Nameserver Eintraege fuer delegierte (fremde) Subdomains

# Auswahl der fremden Zonen (mit Primary Nameserver), die die aktuelle Zone
# als Vaterzone haben
```

```
local(@del_zones);
&query(<<"  END_SQL", *del_zones);
  select id, name, pns_ip_name, pns_ip_addr
  from ${DB}del_zones where parent = $_[11]
END_SQL

# Auswahl der (fremden) Secondary Nameserver einer bestimmten delegierten
# fremden Zone
local($csr_dzs) = &query_open (<<"  END_SQL");
  select sns_ip_name, sns_ip_addr from ${DB}del_zones_sec where del_zone = :1
END_SQL

# Auswahl der lokalen Secondary Nameserver einer bestimmten fremden Zone
local($csr_seclf) = &query_open (<<"  END_SQL");
  select h.name, h.subdomain_name, z.name
  from ${DB}zones z, ${DB}hosts h, ${DB}nameservers n, ${DB}secondary_lf s
  where z.id = h.zone and h.id = n.host and
        n.id = s.nameserver and s.zone_name = :1
END_SQL

# Schleife ueber alle delegierten fremden Subdomains
for(@del_zones)
{
  # Parameter der aktuellen Subdomain
  local($id, $name, $pns_ip_name, $pns_ip_addr) = split(/$;/);
  # $id, $name & $pns_ip_name koennen nicht NULL sein
  # $pns_ip_addr darf u.U. NULL sein
  local($subdomain) = $name;

  # Subdomain-Name (innerhalb) der Domain bestimmen
  $subdomain =~ s/\.$current$//i;

  # Ausgabe des Subdomain-Namens mit Primary Nameserver
  print "$subdomain\tIN\tNS\t$pns_ip_name.\n";

  # ---------------------------------------------------------------------

  # Primary Nameserver Host ist Kandidat fuer Glue-Record,
  # falls sein FQDN in der zur aktuellen Zone gehoerigen Domain liegt
  if($pns_ip_name =~ /\.$current$/i)
  {
    # In diesem Fall muss seine IP-Adresse angegeben werden,
    # sonst Programmabbruch
    &notnull($pns_ip_addr, 'pns_ip_addr', 'del_zones');

    # Loeschen von Leerzeichen in der Adresse (DB-Konvention)
    $pns_ip_addr =~ s/\x20//g;

    # Eintrag in die Liste
    push(@glue, join($;, ($pns_ip_name, '', 'IN', $pns_ip_addr)));
  }
```

```
# ------------------------------------------------------------------

# Fremde Secondary Nameserver
local(@dzs);

# Schleife ueber alle fremden Secondary Nameserver
&query_bind($csr_dzs, *dzs, $id);
for(@dzs)
{
  # Parameter des aktuellen Nameservers
  local($sns_ip_name, $sns_ip_addr) = split(/$;/);
  # $sns_ip_name kann nicht NULL sein
  # $sns_ip_addr darf u.U. NULL sein

  # Ausgabe des Secondary Nameservers
  print "\t\tIN\tNS\t$sns_ip_name.\n";

  # ----------------------------------------------------------------

  # Secondary Nameserver Host ist Kandidat fuer Glue-Record,
  # falls sein FQDN in der zur aktuellen Zone gehoerigen Domain liegt
  if($sns_ip_name =~ /\.$current$/i)
  {
    # In diesem Fall muss seine IP-Adresse angegeben werden,
    # sonst Programmabbruch
    &notnull($sns_ip_addr, 'sns_ip_addr', 'del_zones');

    # Loeschen von Leerzeichen in der Adresse (DB-Konvention)
    $sns_ip_addr =~ s/\x20//g;

    # Eintrag in die Liste
    push(@glue, join($;, ($sns_ip_name, '', 'IN', $sns_ip_addr)));
  }
}

# ------------------------------------------------------------------

# Lokale Secondary Nameserver
local(@seclf);
&query_bind($csr_seclf, *seclf, $name);

# Schleife ueber alle lokalen Secondary Nameserver
for(@seclf)
{
  # Host-Parameter des aktuellen Nameservers
  local($hostname, $subdomainname, $domainname) = split(/$;/);
  # $hostname & $domainname koennen nicht NULL sein
  # $subdomainname darf NULL sein

  # Aufbau des Fully Qualified Domain Name fuer den Secondary Nameserver
```

```
    local($fqdn);
    if((!&testnull($subdomainname)) && (length($subdomainname) > 0))
    {
      $fqdn = $hostname . '.' . $subdomainname . '.' . $domainname;
    }
    else
    {
      $fqdn = $hostname . '.' . $domainname;
    }

    # Ausgabe des Secondary Nameservers
    print "\t\tIN\tNS\t$fqdn.\n";
  }
}

# Schliessen der fuer delegierte (fremde) Subdomains geoeffneten DB-Cursor
&query_close($csr_dzs);
&query_close($csr_seclf);

# ---------------------------------------------------------------------

# Sortieren der Glue-Record Kandidaten, Elimination doppelter Eintraege

local($last) = undef;

# sortierte minimale Liste
local(@sglue) = ();

# Schleife fuer alle Eintraege (innerhalb sind sie bereits sortiert)
for(sort @glue)
{
  # Falls der aktuelle Eintrag schon eingetragen wurde, wird
  # er nicht nochmal eingetragen
  if($_ eq $last) { next; }

  # Eintrag in die neue Liste
  push(@sglue, $_);

  # Merken des aktuellen Eintrags fuer den naechsten Schleifendurchlauf
  $last = $_;
}

# ---------------------------------------------------------------------

# Mail Exchanger (MX-Records) fuer die Zone

# Auswahl der Mail Exchanger Eintraege fuer die aktuelle Zone
local(@mailx);
&query(<<"  END_SQL", *mailx);
  select m.name, m.record_ttl, m.record_class, m.pref_value,
         h.name, h.subdomain_name, z.name
```

```
  from ${DB}zones z, ${DB}hosts h, ${DB}mailx m
  where z.id = h.zone and h.id = m.host and m.zone = $_[11]
END_SQL

# Schleife ueber alle betroffenen Mail Exchanger Eintraege
for(@mailx)
{
  # aktuelle Parameter der Eintraege
  local($name, $record_ttl, $record_class, $pref_value, $hostname,
        $subdomainname, $domainname) = split(/$;/);
  # $name, $pref_value, $hostname, $domainname koennen nicht NULL sein
  # $record_ttl, $record_class, $subdomainname duerfen NULL sein

  # Aufbau des Fully Qualified Domain Name des Ziel-Hosts
  local($fqdn);
  if((!&testnull($subdomainname)) && (length($subdomainname) > 0))
  {
    $fqdn = $hostname . '.' . $subdomainname . '.' . $domainname;
  }
  else
  {
    $fqdn = $hostname . '.' . $domainname;
  }

  # Falls kein bzw. ein leerer Name angegeben wird, dann
  # Eintrag fuer aktuelle Zone
  if((&testnull($name)) || ($name eq '')) { $name = '@'; }

  # Record-TTL und -Class
  if(&testnull($record_ttl))
    { $record_ttl = $def{'MAILX.RECORD_TTL'}; }
  if(&testnull($record_class))
    { $record_class = $def{'MAILX.RECORD_CLASS'}; }

  # Ausgabe des MX-Records
  print "$name\t$record_ttl\t$record_class\tMX\t$pref_value\t$fqdn.\n";
}

# ---------------------------------------------------------------------

# Hosts (A-, CNAME-, HINFO- & WKS-Records) fuer die Zone

# Auswahl der Hosts in der aktuellen Zone
local(@hosts);
&query(<<"  END_SQL", *hosts);
  select id, name, subdomain_name, record_ttl, record_class, cpu, os
  from ${DB}hosts
  where zone = $_[11]
END_SQL

# Auswahl der WKS eines bestimmten Hosts
```

```
local($csr_wks) = &query_open(<<"  END_SQL");
  select service from ${DB}wks where host = :1
END_SQL

# Auswahl der Alias-Namen eines bestimmten Hosts
local($csr_alias) = &query_open(<<"  END_SQL");
  select canonical_name from ${DB}aliases where host = :1
END_SQL

# ---------------------------------------------------------------------

# Schleife ueber alle Hosts in der aktuellen Zone
for(@hosts)
{
  # akutelle Host-Parameter
  local($id, $name, $subdomainname, $record_ttl, $record_class,
        $cpu, $os) = split(/$;/);
  # $id, $name koennen nicht NULL sein
  # $subdomainname, $record_ttl, $record_class, $cpu & $os duerfen NULL sein

  # Name innerhalb der aktuellen Zone
  local($hostname) = $name;
  if((!&testnull($subdomainname)) && (length($subdomainname) > 0))
  {
    $hostname .= '.' . $subdomainname;
  }

  # Record-TTL und -Class
  if(&testnull($record_ttl))
    { $record_ttl = $def{'HOSTS.RECORD_TTL'}; }
  if(&testnull($record_class))
    { $record_class = $def{'HOSTS.RECORD_CLASS'}; }

  # Falls fuer einen Host keine einzige IP-Adresse vorhanden ist, ...
  if(&query_bind($csr_ipaddr, *ipaddr, $id) == 0)
  {
    # ..., dann Programmabbruch
    die "No IP-adress available for $hostname.$current";
  }

  # Auswahl der ersten IP-Adresse
  # $ipaddr[..] kann nicht NULL sein
  local($ipa) = shift(@ipaddr);

  # Loeschen von Leerzeichen in der Adresse (DB-Konvention)
  $ipa =~ s/\x20//g;

  # Ausgabe des Hostnamens (innerhalb der aktuellen Zone) und der
  # (ersten) IP-Adresse als A-Record
  print "$hostname\t$record_ttl\t$record_class\tA\t$ipa\n";
```

```
# Schleife ueber restliche IP-Adressen des Hosts
for(@ipaddr)
{
  # Loeschen von Leerzeichen in der Adresse (DB-Konvention)
  s/\x20//g;

  # Ausgabe des entsprechenden A-Records
  print "\t$record_ttl\t$record_class\tA\t$_\n";
}

# Erscheint der Host in der Glue-Liste, muss er dort geloescht
# (fuer ungueltig erklaert) werden (sonst doppelter Eintrag)
for(@sglue)
{
  if(/^$hostname\.$current$;/i) { $_ = undef; last; }
}

# ---------------------------------------------------------------

# Existieren fuer den Host gueltige Angaben ueber CPU & Betriebssystem, ...
if((!&testnull($cpu)) && (length($cpu) > 0) &&
   (!&testnull($os)) && (length($os) > 0))
{
  # ..., so werden diese als HINFO-Record ausgegeben
  print "\t$record_ttl\t$record_class\tHINFO\t$cpu $os\n";
}

# ---------------------------------------------------------------

# WKS des Hosts
local(@wks);
&query_bind($csr_wks, *wks, $id);
# $wks[..] kann nicht NULL sein

# Schleife ueber alle Eintraege
for(@wks)
{
  # Ausgabe als WKS-Record
  print "\t$record_ttl\t$record_class\tWKS\t$_\n";
}

# ---------------------------------------------------------------

# Aliasnamen des Hosts
local(@alias);
&query_bind($csr_alias, *alias, $id);
# $alias[..] kann nicht NULL sein

# Schleife ueber alle Alias-Namen
for(@alias)
{
```

```
      # ACHTUNG: Alias-Namen werden nicht direkt der Zone zugeordnet,
      # sondern der jeweiligen Subdomain des Hosts
      if((!&testnull($subdomainname)) && (length($subdomainname) > 0))
      {
        $_ .= '.' . $subdomainname;
      }

      # Ausgabe als CNAME-Record
      print "$_\t$record_ttl\t$record_class\tCNAME\t$hostname\n";
    }
  }

  # Schliessen der fuer die Hostausgabe geoeffneten DB-Cursor
  &query_close($csr_wks);
  &query_close($csr_alias);
  &query_close($csr_ipaddr);

  # ------------------------------------------------------------------

  # Erzeugen der Glue-Records

  # Schleife ueber alle Listeneintraege
  for(@sglue)
  {
    # Wurde der aktuelle Eintrag geloescht (fuer ungueltig) erklaert,
    # so darf er nicht weiter beruecksichtigt werden
    if(defined($_))
    {
      # aktuelle Glueparameter
      local($fqdn, $record_ttl, $record_class, $ipa) = split(/$;/);

      # Falls der Host in der zur aktuellen Zone gehoerenden Domain liegt, ...
      if($fqdn =~ s/\.$current$//i)
      {
        # ..., muss er als A-Record (Glue) eingetragen werden
        print "$fqdn\t$record_ttl\t$record_class\tA\t$ipa\n";
      }
    }
  }
}

# ********************************************************************
1;
```

A.8 Modul named_reverse.pl

```
# ************************************************************************
# * Integration des Managements des Domain-Name-Systems in die           *
# * vorhandene Managementumgebung des Leibniz-Rechenzentrums (LRZ)       *
# * (Diplomarbeit an der TU-Muenchen, SS 94 - WS 94/95)                  *
# * -------------------------------------------------------------------- *
# * Autor: Heiko Hauck (hauck@informatik.tu-muenchen.de)                 *
# * Datei: named_reverse.pl (Oraperl Bibliothek fuer ora2dns)            *
# ************************************************************************

# PROZEDUR zur Generierung einer Zonen-Datei fuer das Reverse Mapping
# Parameter: $_[0]  = Name der Datei
#            $_[1]  = FQDN des Nameserverhosts
#            $_[2]  = IP-Adresse des Nameserverhosts
#            $_[3]  = Klasse des SOA-Records
#            $_[4]  = E-Mail-Adresse (SOA-Responsible)
#            $_[5]  = Seriennummer im SOA-Record
#            $_[6]  = Refresh-Intervall im SOA-Record
#            $_[7]  = Retry-Intervall im SOA-Record
#            $_[8]  = Expire-Intervall im SOA-Record
#            $_[9]  = Minimum-Intervall im SOA-Record
#            $_[10] = Domainname der Zone
#            $_[11] = Identifikator der Zone
# Die Ausgabe erfolgt zur Standard-Ausgabe.

sub named_reverse
{
  # Ausgabe des Dateikopfes
  print "; $_[0] for $_[1] [$_[2]]\n";
  print "; created by ora2dns (08/94 by Heiko Hauck; Diplomarbeit TUM)\n";
  print ";\n";
  print "; User: ", `whoami`;
  print "; Date: ", `date`;
  print ";\n";

  # Ausgabe der betroffenen Zone
  print "\$ORIGIN\t\t$_[10].\n";

  # Ausgabe des SOA-Records fuer diese Zone
  print "\@\t\t$_[3]\tSOA\t$_[1]. $_[4]. (\n";
  print "\t\t\t\t\t$_[5]   ; serial\n";
  print "\t\t\t\t\t$_[6]   ; refresh\n";
  print "\t\t\t\t\t$_[7]   ; retry\n";
  print "\t\t\t\t\t$_[8]   ; expire\n";
  print "\t\t\t\t\t$_[9] ) ; minimum\n";

  # ---------------------------------------------------------------------

  # Achtung: 'Perl' ueberschreibt aktuelle Parameter ohne entspr. Zuweisung!
  # Deshalb: Kopie in lokale Variable
```

```
local($current) = $_[10];

# ---------------------------------------------------------------------

# Aufbau der Maske zur Auswahl der IP-Adressen
# Dabei wird unterschieden, wieviele Bytes der IP-Adresse durch den
# Domainnamen schon vorgegeben sind

local($mask, $replace);
if($current =~ /^(\d{1,3})\.(\d{1,3})\.(\d{1,3})\.in-addr\.arpa$/i)
{
  # 3 Bytes vorgegeben
  # Beispiel: Fuer die Zone '15.187.129.in-addr.arpa' ist die zugehoerige
  # Maske '129.187. 15.'
  # Diese wird spaeter mit dem Begin der IP-Adressen verglichen
  $mask = ' ' x (3 - length($3)) . $3 . '.' .
          ' ' x (3 - length($2)) . $2 . '.' .
          ' ' x (3 - length($1)) . $1 . '.';

  # 1 Byte ist noch auszuwaehlen
  $replace = 1;
}
else
{
  if($current =~ /^(\d{1,3})\.(\d{1,3})\.in-addr\.arpa$/i)
  {
    # 2 Bytes vorgegeben
    $mask = ' ' x (3 - length($2)) . $2 . '.' .
            ' ' x (3 - length($1)) . $1 . '.';

    # 2 Bytes sind noch auszuwaehlen
    $replace = 2;
  }
  else
  {
    if($current =~ /^(\d{1,3})\.in-addr\.arpa$/i)
    {
      # 1 Byte vorgegeben
      $mask = ' ' x (3 - length($1)) . $1 . '.';

      # 3 Bytes sind noch auszuwaehlen
      $replace = 3;
    }
    else
    {
      if($current =~ /^\.in-addr\.arpa$/i)
      {
        # kein Byte vorgegeben
        $mask = '';

        # alle 4 Bytes sind noch auszuwaehlen
```

```
        $replace = 4;
      }
      else
      {
        # Programmabbruch, falls ein fehlerhafter Domainname angegeben wurde
        die "Illegal domain $current";
      }
    }
  }
}

# Liste von Subnetzmasken, deren IP-Adressen nicht zu dieser,
# sondern zu einer delegierten Zone gehoeren (siehe unten)
local(@submasks) = ();

# ---------------------------------------------------------------------

# Eintrag fuer den Primary Nameserver der Zone
print "\t\tIN\tNS\t$_[1].\n";

# ---------------------------------------------------------------------

# Lokale Secondary Nameserver fuer die Zone
local(@servers);
&query(<<"  END_SQL", *servers);
  select h.name, h.subdomain_name, z.name
  from ${DB}zones z, ${DB}hosts h, ${DB}nameservers n, ${DB}secondary_ll s
  where z.id = h.zone and h.id = n.host and
        n.id = s.nameserver and s.zone = $_[11]
END_SQL

# Schleife ueber alle betroffenen Nameserver
for(@servers)
{
  # Parameter des aktuellen Nameservers
  local($hostname, $subdomainname, $domainname) = split(/$;/);
  # $hostname, $domainname koennen nicht NULL sein
  # $subdomainname darf NULL sein

  # Aufbau des Fully Qualified Domain Name fuer den Nameserver-Host
  local($fqdn);
  if((!&testnull($subdomainname)) && (length($subdomainname) > 0))
  {
    $fqdn = $hostname . '.' . $subdomainname . '.' . $domainname;
  }
  else
  {
    $fqdn = $hostname . '.' . $domainname;
  }

  # Eintrag fuer den Nameserver
```

```
  print "\t\tIN\tNS\t$fqdn.\n";
}

# ---------------------------------------------------------------------

# Fremde Secondary Nameserver fuer die Zone

# Auswahl der FQDNs der betroffenen Nameserver
&query(<<"  END_SQL", *servers);
  select ip_name from ${DB}secondary_fl where zone = $_[11]
END_SQL

# Schleife ueber alle betroffenen Nameserver
for(@servers)
{
  # $_ kann nicht NULL sein

  # Eintrag fuer den Nameserver
  print "\t\tIN\tNS\t$_.\n";
}

# ---------------------------------------------------------------------

# Nameserver Eintraege fuer delegierte (lokale) Subdomains

# Auswahl aller Zonen, die die aktuelle Zone als Vaterzone haben
local(@domains);
&query(<<"  END_SQL", *domains);
  select id, name, prim_nameserver from ${DB}zones where parent = $_[11]
END_SQL

# Auswahl des Namens, Subdomain- und Domainnamens eines Hosts,
# auf dem ein bestimmter Primary Nameserver laeuft
local($csr_prim) = &query_open(<<"  END_SQL");
  select h.name, h.subdomain_name, z.name
  from ${DB}zones z, ${DB}hosts h, ${DB}nameservers n
  where z.id = h.zone and h.id = n.host and n.id = :1
END_SQL

# Auswahl des Namens, Subdomain- und Domainnamens von Hosts,
# auf denen fuer eine bestimmte Zone Secondary Nameserver laufen
local($csr_serv) = &query_open(<<"  END_SQL");
  select h.name, h.subdomain_name, z.name
  from ${DB}zones z, ${DB}hosts h, ${DB}nameservers n, ${DB}secondary_ll s
  where z.id = h.zone and h.id = n.host and
        n.id = s.nameserver and s.zone = :1
END_SQL

# Auswahl von fremden Secondary Nameservern fuer eine bestimmte Zone
local($csr_secf) = &query_open (<<"  END_SQL");
  select ip_name from ${DB}secondary_fl where zone = :1
```

```
END_SQL

# ------------------------------------------------------------------

# Schleife ueber alle delegierten lokalen Subdomains
for(@domains)
{
  # Parameter der akuellen Subdomain
  local($id, $name, $prim_nameserver) = split(/$;/);
  # $id, $name & $prim_nameserver koennen nicht NULL sein

  # Subdomain-Name (innerhalb) der Domain bestimmen
  local($subdomain) = $name;
  $subdomain =~ s/\.$current$//i;

  # Primary Nameserver fuer Subdomain
  local(@prim);
  &query_bind($csr_prim, *prim, $prim_nameserver);
  local($hostname, $subdomainname, $domainname) = split(/$;/, $prim[0]);
  # $hostname, $domainname koennen nicht NULL sein
  # $subdomainname darf NULL sein

  # Aufbau des Fully Qualified Domain Name fuer den Primary Nameserver
  local($fqdn);
  if((!&testnull($subdomainname)) && (length($subdomainname) > 0))
  {
    $fqdn = $hostname . '.' . $subdomainname . '.' . $domainname;
  }
  else
  {
    $fqdn = $hostname . '.' . $domainname;
  }

  # Ausgabe des Subdomain-Namens mit Primary Nameserver
  print "$subdomain\t\tIN\tNS\t$fqdn.\n";

  # ------------------------------------------------------------------

  # Aufbau der Subnetzmaske zum Ausschluss von IP-Adressen, die in
  # die lokale delegierte Subdomain fallen

  local($submask);

  if($subdomain =~ /^(\d{1,3})\.(\d{1,3})\.(\d{1,3})$/i)
  {
    # 3 Bytes vorgegeben

    # nur moeglich, wenn noch 4 Bytes zu besetzen sind,
    # sonst Programmabbruch
    if($replace < 4)
      { die "Illegal subdomain $subdomain of domain $current"; }
```

```
  $submask = ' ' x (3 - length($3)) . $3 . '.' .
             ' ' x (3 - length($2)) . $2 . '.' .
             ' ' x (3 - length($1)) . $1 . '.';
}
else
{
  if($subdomain =~ /^(\d{1,3})\.(\d{1,3})$/i)
  {
    # 2 Bytes vorgegeben

    # nur moeglich, wenn noch mindestens 3 Bytes zu besetzen sind,
    # sonst Programmabbruch
    if($replace < 3)
      { die "Illegal subdomain $subdomain of domain $current"; }

    $submask = ' ' x (3 - length($2)) . $2 . '.' .
               ' ' x (3 - length($1)) . $1 . '.';
  }
  else
  {
    if($subdomain =~ /^(\d{1,3})$/i)
    {
      # 1 Byte vorgegeben

      # nur moeglich, wenn noch mindestens 2 Bytes zu besetzen sind,
      # sonst Programmabbruch
      if($replace < 2)
        { die "Illegal subdomain $subdomain of domain $current"; }

      $submask = ' ' x (3 - length($1)) . $1 . '.';
    }
    else
    {
      # nicht zulaessige Subdomain, Programmabbruch
      die "Illegal subdomain $subdomain of domain $current";
    }
  }
}

# Eintrag der Subnetzmaske in die dafuer vorgesehene Liste
push(@submasks,$submask);

# ------------------------------------------------------------------

# Lokale Secondary Nameserver fuer die Subdomain
local(@serv);
&query_bind($csr_serv, *serv, $id);

# Schleife ueber alle lokalen Secondary Nameserver
for(@serv)
```

```
  {
    # Host-Parameter des aktuellen Nameservers
    local($hostname, $subdomainname, $domainname) = split(/$;/);
    # $hostname & $domainname koennen nicht NULL sein
    # $subdomainname darf NULL sein

    # Aufbau des Fully Qualified Domain Name fuer den Secondary Nameserver
    local($fqdn);
    if((!&testnull($subdomainname)) && (length($subdomainname) > 0))
    {
      $fqdn = $hostname . '.' . $subdomainname . '.' . $domainname;
    }
    else
    {
      $fqdn = $hostname . '.' . $domainname;
    }

    # Ausgabe des Secondary Nameservers
    print "\t\tIN\tNS\t$fqdn.\n";
  }

  # ---------------------------------------------------------------------

  # Fremde Secondary Nameserver fuer die Subdomain
  local(@secf);
  &query_bind($csr_secf, *secf, $id);

  # Schleife ueber alle fremden Secondary Nameserver
  for(@secf)
  {
    # $_ kann nicht NULL sein

    # Ausgabe des Secondary Nameservers
    print "\t\tIN\tNS\t$_.\n";
  }
}

# Schliessen der fuer delegierte (lokale) Subdomains geoeffneten DB-Cursor
&query_close($csr_prim);
&query_close($csr_serv);
&query_close($csr_secf);

# ---------------------------------------------------------------------

# Nameserver Eintraege fuer delegierte (fremde) Subdomains

# Auswahl der fremden Zonen (mit Primary Nameserver), die die aktuelle Zone
# als Vaterzone haben
local(@del_zones);
&query(<<"  END_SQL", *del_zones);
  select id, name, pns_ip_name
```

```
  from ${DB}del_zones where parent = $_[11]
END_SQL

# Auswahl der (fremden) Secondary Nameserver einer bestimmten delegierten
# fremden Zone
local($csr_dzs) = &query_open (<<"  END_SQL");
  select sns_ip_name from ${DB}del_zones_sec where del_zone = :1
END_SQL

# Auswahl der lokalen Secondary Nameserver einer bestimmten fremden Zone
local($csr_seclf) = &query_open (<<"  END_SQL");
  select h.name, h.subdomain_name, z.name
  from ${DB}zones z, ${DB}hosts h, ${DB}nameservers n, ${DB}secondary_lf s
  where z.id = h.zone and h.id = n.host and
        n.id = s.nameserver and s.zone_name = :1
END_SQL

# Schleife ueber alle delegierten fremden Subdomains
for(@del_zones)
{
  # Parameter der aktuellen Subdomain
  local($id, $name, $pns_ip_name,) = split(/$;/);
  # $id, $name & $pns_ip_name koennen nicht NULL sein

  # Subdomain-Name (innerhalb) der Domain bestimmen
  local($subdomain) = $name;
  $subdomain =~ s/\.$current$//i;

  # Ausgabe des Subdomain-Namens mit Primary Nameserver
  print "$subdomain\tIN\tNS\t$pns_ip_name.\n";

  # Fremde Secondary Nameserver
  local(@dzs);
  &query_bind($csr_dzs, *dzs, $id);

  # Schleife ueber alle fremden Secondary Nameserver
  for(@dzs)
  {
    # $_ kann nicht NULL sein

    # Ausgabe des Secondary Nameservers
    print "\t\tIN\tNS\t$_.\n";
  }

  # Lokale Secondary Nameserver
  local(@seclf);
  &query_bind($csr_seclf, *seclf, $name);

  # Schleife ueber alle lokalen Secondary Nameserver
  for(@seclf)
  {
```

```
    # Host-Parameter des aktuellen Nameservers
    local($hostname, $subdomainname, $domainname) = split(/$;/);
    # $hostname & $domainname koennen nicht NULL sein
    # $subdomainname darf NULL sein

    # Aufbau des Fully Qualified Domain Name fuer den Secondary Nameserver
    local($fqdn);
    if((!&testnull($subdomainname)) && (length($subdomainname) > 0))
    {
      $fqdn = $hostname . '.' . $subdomainname . '.' . $domainname;
    }
    else
    {
      $fqdn = $hostname . '.' . $domainname;
    }

    # Ausgabe des Secondary Nameservers
    print "\t\tIN\tNS\t$fqdn.\n";
  }
}

# Schliessen der fuer delegierte (fremde) Subdomains geoeffneten DB-Cursor
&query_close($csr_dzs);
&query_close($csr_seclf);

# ------------------------------------------------------------------

# PTR-Eintraege fuer die Zone

# Auswahl saemtlicher Hosts (Name, Subdomain- & Domainname, IP-Adresse),
# bei denen eine IP-Adresse zu der Maske passt
local(@ptr);
&query(<<"  END_SQL", *ptr);
  select h.name, h.subdomain_name, z.name, i.ip_addr
  from ${DB}ipaddr i, ${DB}hosts h, ${DB}zones z
  where z.id = h.zone and h.id = i.host and i.ip_addr like '$mask\%'
END_SQL

# Schleife ueber alle ausgewaehlten Hosts
PTR: for(@ptr)
{
  # Parameter des aktuellen Hosts
  local($hostname, $subdomainname, $domainname, $ipa) = split(/$;/);
  # $hostname, $domainname & $ipa koennen nicht NULL sein
  # $subdomainname darf NULL sein

  # Ausschluss von IP-Adressen, die einer delegierten Zone (Subnetz)
  # angehoeren
  for(@submasks)
  {
    if($ipa =~ /^$mask$_/) {next PTR;}
```

```
    }

    # Aufbau des Fully Qualified Domain Name
    local($fqdn);
    if((!&testnull($subdomainname)) && (length($subdomainname) > 0))
    {
      $fqdn = $hostname . '.' . $subdomainname . '.' . $domainname;
    }
    else
    {
      $fqdn = $hostname . '.' . $domainname;
    }

    # Zerlegen der IP-Adresse in ihre 4 Bytes
    $ipa =~ /^\x20*(\d+)\.\x20*(\d+)\.\x20*(\d+)\.\x20*(\d+)$/;

    # Das 4. Byte wird immer ausgegeben
    print $4;

    # Byte 1-3 werden nur ausgegeben, wenn deren Auswahl innerhalb
    # der Zone noch frei war
    if($replace >= 2) { print ".$3"; }
    if($replace >= 3) { print ".$2"; }
    if($replace == 4) { print ".$1"; }

    # Ausgabe des restlichen Records (Klasse, Typ, FQDN)
    print "\t\tIN\tPTR\t$fqdn.\n";
  }
}

# ****************************************************************************
1;
```

A.9 Modul named_local.pl

```
# ************************************************************************
# * Integration des Managements des Domain-Name-Systems in die           *
# * vorhandene Managementumgebung des Leibniz-Rechenzentrums (LRZ)       *
# * (Diplomarbeit an der TU-Muenchen, SS 94 - WS 94/95)                  *
# * -------------------------------------------------------------------- *
# * Autor: Heiko Hauck (hauck@informatik.tu-muenchen.de)                 *
# * Datei: named_local.pl (Oraperl Bibliothek fuer ora2dns)              *
# ************************************************************************

# PROZEDUR zur Generierung der Zonen-Datei fuer das Loopback-Interface
# (127.1 -> localhost)
# Parameter: $_[0] = Name der Datei
#            $_[1] = IP-Adresse des Nameserverhosts
#            $_[2] = Klasse des SOA-Records
#            $_[3] = FQDN des Nameserverhosts
#            $_[4] = E-Mail-Adresse (SOA-Responsible)
#            $_[5] = Seriennummer im SOA-Record
#            $_[6] = Refresh-Intervall im SOA-Record
#            $_[7] = Retry-Intervall im SOA-Record
#            $_[8] = Expire-Intervall im SOA-Record
#            $_[9] = Minimum-Intervall im SOA-Record
# Die Ausgabe erfolgt zur Standard-Ausgabe.

sub named_local
{
  # Ausgabe des Dateikopfes
  print "; $_[0] for $_[3] [$_[1]]\n";
  print "; created by ora2dns (08/94 by Heiko Hauck; Diplomarbeit TUM)\n";
  print ";\n";
  print "; User: ", `whoami`;
  print "; Date: ", `date`;
  print ";\n";

  # Ausgabe der betroffenen Zone
  print "\$ORIGIN\t0.0.127.in-addr.arpa.\n";

  # Ausgabe des SOA-Records fuer diese Zone
  print "\@\t$_[2]\tSOA\t$_[3]. $_[4]. (\n";
  print "\t\t\t\t$_[5]   ; serial\n";
  print "\t\t\t\t$_[6]   ; refresh\n";
  print "\t\t\t\t$_[7]   ; retry\n";
  print "\t\t\t\t$_[8]   ; expire\n";
  print "\t\t\t\t$_[9] ) ; minimum\n";

  # Ausgabe des Nameservers fuer diese Zone (= aktueller Nameserver)
  print "\tIN\tNS\t$_[3].\n";

  # Ausgabe des PTR-Records, der fuer die eigentliche Abbildung zustaendig ist
  print "1\tIN\tPTR\tlocalhost.\n";
```

```
}

# ************************************************************************
1;
```

A.10 Modul named_loopback.pl

```
# ************************************************************************
# * Integration des Managements des Domain-Name-Systems in die         *
# * vorhandene Managementumgebung des Leibniz-Rechenzentrums (LRZ)     *
# * (Diplomarbeit an der TU-Muenchen, SS 94 - WS 94/95)                *
# * ------------------------------------------------------------------ *
# * Autor: Heiko Hauck (hauck@informatik.tu-muenchen.de)               *
# * Datei: named_loopback.pl (Oraperl Bibliothek fuer ora2dns)         *
# ************************************************************************

# PROZEDUR zur Generierung der Zonen-Datei fuer das Loopback-Interface
# (localhost -> 127.1)
# Parameter: $_[0] = Name der Datei
#            $_[1] = IP-Adresse des Nameserverhosts
#            $_[2] = Klasse des SOA-Records
#            $_[3] = FQDN des Nameserverhosts
#            $_[4] = E-Mail-Adresse (SOA-Responsible)
#            $_[5] = Seriennummer im SOA-Record
#            $_[6] = Refresh-Intervall im SOA-Record
#            $_[7] = Retry-Intervall im SOA-Record
#            $_[8] = Expire-Intervall im SOA-Record
#            $_[9] = Minimum-Intervall im SOA-Record
# Die Ausgabe erfolgt zur Standard-Ausgabe.

sub named_loopback
{
  # Ausgabe des Dateikopfes
  print "; $_[0] for $_[3] [$_[1]]\n";
  print "; created by ora2dns (08/94 by Heiko Hauck; Diplomarbeit TUM)\n";
  print ";\n";
  print "; User: ", `whoami`;
  print "; Date: ", `date`;
  print ";\n";

  # Ausgabe der betroffenen Zone
  print "\$ORIGIN\tlocalhost.\n";

  # Ausgabe des SOA-Records fuer diese Zone
  print "\@\t$_[2]\tSOA\t$_[3]. $_[4]. (\n";
  print "\t\t\t\t$_[5]   ; serial\n";
  print "\t\t\t\t$_[6]   ; refresh\n";
  print "\t\t\t\t$_[7]   ; retry\n";
  print "\t\t\t\t$_[8]   ; expire\n";
```

```
  print "\t\t\t\t$_[9] ) ; minimum\n";

  # Ausgabe des Nameservers fuer diese Zone (= aktueller Nameserver)
  print "\tIN\tNS\t$_[3].\n";

  # Ausgabe des A-Records, der fuer die eigentliche Abbildung zustaendig ist
  print "\tIN\tA\t127.0.0.1\n";
}

# ****************************************************************************
1;
```

A.11 Modul named_cache.pl

```
# ************************************************************************
# * Integration des Managements des Domain-Name-Systems in die           *
# * vorhandene Managementumgebung des Leibniz-Rechenzentrums (LRZ)       *
# * (Diplomarbeit an der TU-Muenchen, SS 94 - WS 94/95)                  *
# * -------------------------------------------------------------------- *
# * Autor: Heiko Hauck (hauck@informatik.tu-muenchen.de)                 *
# * Datei: named_cache.pl (Oraperl Bibliothek fuer ora2dns)              *
# ************************************************************************

# PROZEDUR zum Holen der Root-Domain-Datei (Cache) mittels Anonymous FTP
# Parameter: $_[0] = Name der lokalen Datei (inkl. Verzeichnispfad)
#            $_[1] = Name der FTP-Hosts, von dem die Datei geholt wird
#            $_[2] = Verzeichnis, in dem dort die Datei abgelegt ist
#            $_[3] = Name der Datei auf dem FTP-Host
#            $_[4] = E-Mail-Adresse (als Passwort fuer Anonymous FTP)

sub named_cache
{
  # lokale Parameterkopien
  local($local_file, $host, $remote_dir, $remote_file, $email) = @_;

  # Die Login-Name- & Passwort-Uebergabe an das FTP geschieht mittels
  # der '.netrc'-Datei des Anwenders.
  # Ist in dieser Datei noch kein ein Eintrag fuer den angegebenen FTP-Host
  # vorhanden, so wird er eingefuegt (ein vorhandener wird nicht geaendert).
  # Ist die Datei noch nicht vorhanden, wird sie mit dem Eintrag erzeugt.

  # '.netrc' liegt im Home-Verzeichnis des Anwenders
  local($filename) = $ENV{'HOME'}.'/.netrc';

  # Eintrag in '.netrc'
  local($netrc) = "machine $host login ftp password $email\n";

  # Existiert die Datei '.netrc' bereits?
  if(-e $filename)
  {
    # Datei oeffnen zum Lesen
    # Misslingt dies, dann Programmabbruch
    open(NETRC, "<$filename") || die "Can't open $filename: $!\n";

    # Zeilenweises Durchsuchen nach einen entspr. Host-Eintrag
    while(<NETRC>)
    {
      if(/^machine\s+$host\s+/)
      {
        # Wird ein solcher gefunden, braucht kein weiterer mehr gesucht werden
        $netrc = undef; # Markierung fuer "gefunden"
        last;
      }
```

```
  }

  # Datei (vom Lesen) schliessen
  close(NETRC);

  # Falls noch kein entsprechender Eintrag vorhanden ist:
  if(defined($netrc))
  {
    # Datei oeffnen zum Anfuegen
    # Misslingt dies, dann Programmabbruch
    open(NETRC, ">>$filename") || die "Can't append to $filename: $!\n";

    # Eintrag
    print NETRC $netrc;

    # Datei (vom Anfuegen) schliessen
    close(NETRC);
  }
}

else
# Datei '.netrc' existiert noch nicht
{
  # Datei oeffnen zum Schreiben
  # Misslingt dies, dann Programmabbruch
  open(NETRC, ">$filename") || die "Can't write to $filename: $!\n";

  # Eintrag
  print NETRC $netrc;

  # Datei (vom Schreiben) schliessen
  close(NETRC);
}

# Zugriffsrechte auf die Datei ".netrc" (notwendig)
chmod 0600, $filename;

# ------------------------------------------------------------------

# Oeffnen eines Ausgabekanals, der als Eingabe fuer FTP dient
open(FTP,"|ftp $host");

# ASCII-Modus, da eine Text-Datei uebertragen wird
print FTP "ascii\n";

# Wahl des Verzeichnisses auf dem FTP-Host
print FTP "cd $remote_dir\n";

# Datei soll geholt werden
print FTP "get\n";
```

```
  # Angabe des Dateinamens auf dem FTP-Host
  print FTP "$remote_file\n";

  # Angabe des lokalen Dateinamens (inkl. Verzeichnis)
  print FTP "$local_file\n";

  # FTP verlassen
  print FTP "quit\n";

  # Schliessen des Kanals und Durchfuehrung der FTP-Prozedur
  close(FTP);

  print "\n";
}

# ****************************************************************************
1;
```

Anhang B

Datenbank – Beispiel

ALIASES

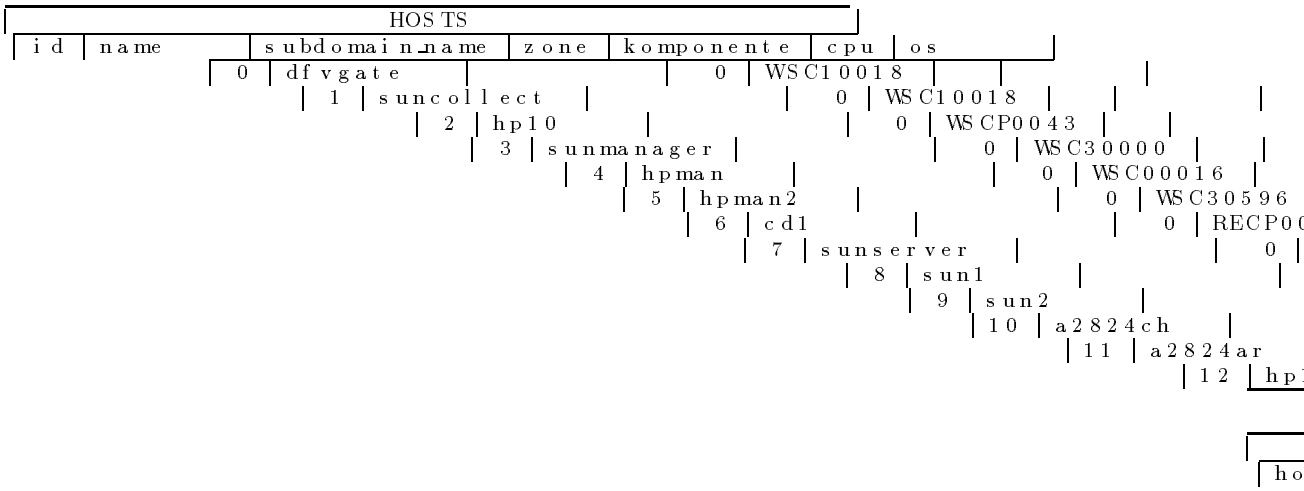
host	canonical_name	*Kommentar*
0	nettest	dfvgate
0	dfv	
0	ircserver	
3	x500	sunmanager
3	ntp1	
4	hpmanager	hpman
4	hpmgr	
4	ntp2	
6	mailhost	cd1
7	mlists	sunserver
7	ftp	
7	lrznews	
7	news	
7	idserver	
7	listserv	
7	gopher	
8	sunpop	sun1

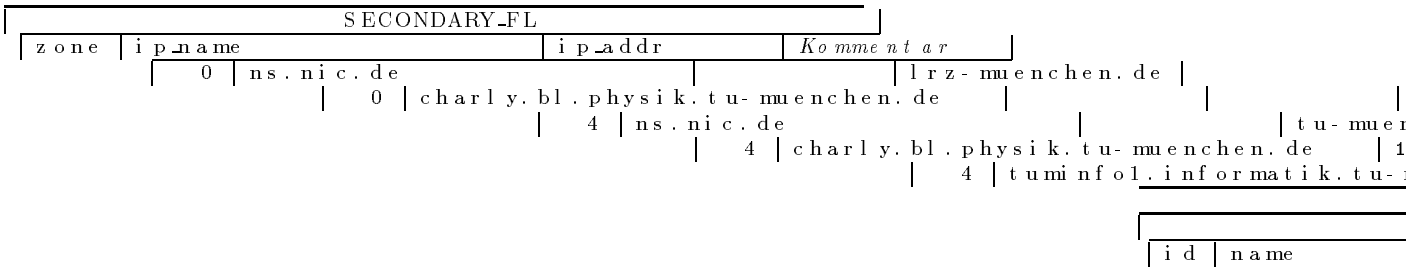
DEL_ZONES

id	name	pns_ip_name	pns_ip_addr
0	informatik.tu-muenchen.de	tuminfo1.informatik.tu-muenchen.de	131.159

DEL_ZONES_SEC

del_zone	sns_ip_name	sns_ip_addr
0	tuminfo2.informatik.tu-muenchen.de	131.159.␣␣0.␣81
0	hpsystem1.informatik.tu-muenchen.de	131.159.␣␣0.176
0	ns.Germany.EU.net	


HOSTS
id
name
subdomain_name
zone
komponente
cpu
os
0 dfvgate 0 WSC10018
1 suncollect 0 WSC10018
2 hp10 0 WSCP0043
3 sunmanager 0 WSC30000
4 hpman 0 WSC00016
5 hpman2 0 WSC30596
6 cd1 0 RECP00
7 sunserver 0
8 sun1
9 sun2
10 a2824ch
11 a2824ar
12 hp


SECONDARY_FL
zone
ip_name
ip_addr
Kommentar
0
ns.nic.de
lrz-muenchen.de
0
charly.bl.physik.tu-muenchen.de
4
ns.nic.de
tu-muen
4
charly.bl.physik.tu-muenchen.de
1
4
tuminfo1.informatik.tu-
id
name

Anhang C

Generierte Konfigurations – Beispiele

Die im folgenden angegebenen Dat
[...] bzw. vom Layout an die
werden dadurch nicht beeinf

C.1 Resolverkonfigurati

```
# /etc/resolv.conf for dfvgate.lrz-muenchen
# created by ora2dns (08/94 by Heiko Hauck;

# User: a2824ch
# Date: Wed Oct 26 15:42:48 MET 1994

domain     lrz-muenchen.de

nameserver 127.0.0.1           # localhost
nameserver 129.187.13.22       # hp10.lrz-m
```

C.2 Nameserverkonfiguration

```
; /etc/named.boot for dfvgate.lrz-muenchen.de [129.187.10.25]
; created by ora2dns (08/94 by Heiko Hauck; Diplomarbeit TUM)
;
; User: a2824ch
; Date: Wed Oct 26 15:42:56 MET 1994
;
; type          domain                  source file or host
```

```
;
directory       /etc/namedb
; primary nameserver:
primary         lrz-muenchen.de         lrz-muenchen.de.DNF
primary         ppp.lrz-muenchen.de     ppp.lrz-muenchen.de.DNF
primary         187.129.in-addr.arpa    187.129.in-addr.arpa.DNF
primary         15.187.129.in-addr.arpa 15.187.129.in-addr.arpa.DNF
primary         tu-muenchen.de          tu-muenchen.de.DNF
primary         0.0.127.in-addr.arpa    named.local
primary         localhost               named.loopback
; secondary nameserver:
secondary       informatik.tu-muenchen.de 131.159.0.1   informatik.tu-muenchen.de.BF
secondary       eg                      193.227.1.1     eg.BF
; cache nameserver:
cache           .                       named.cache
```

C.3 Zonenkonfiguration (Forward Mapping)

```
; /etc/namedb/lrz-muenchen.de.DNF for dfvgate.lrz-muenchen.de [129.187.10.25]
; created by ora2dns (08/94 by Heiko Hauck; Diplomarbeit TUM)
;
; User: a2824ch
; Date: Wed Oct 26 15:42:56 MET 1994
;
$ORIGIN         lrz-muenchen.de.
@               IN      SOA     dfvgate.lrz-muenchen.de. postmaster@[...] (
                                1994082900   ; serial
                                21600        ; refresh
                                3600         ; retry
                                604800       ; expire
                                172800 )     ; minimum
                IN      NS      dfvgate.lrz-muenchen.de.
                IN      NS      hp10.lrz-muenchen.de.
                IN      NS      ns.nic.de.
                IN      NS      charly.bl.physik.tu-muenchen.de.
ppp             IN      NS      dfvgate.lrz-muenchen.de.
                IN      NS      hp10.lrz-muenchen.de.
@               IN      MX      140     cd1.lrz-muenchen.de.
sunmail         IN      MX      10      sunserver.lrz-muenchen.de.
hpmail          IN      MX      10      hp10.lrz-muenchen.de.
dfvgate         IN      A       129.187.10.25
nettest         IN      CNAME   dfvgate
dfv             IN      CNAME   dfvgate
ircserver       IN      CNAME   dfvgate
suncollect      IN      A       129.187.9.241
hp10            IN      A       129.187.13.22
                IN      A       129.187.16.1
sunmanager      IN      A       129.187.10.32
x500            IN      CNAME   sunmanager
```

```
ntp1            IN      CNAME   sunmanager
[...]
sun1            IN      A       129.187.10.20
sunpop          IN      CNAME   sun1
sun2            IN      A       129.187.10.21
a2824ar.ppp     IN      A       129.187.15.157
                IN      HINFO   PC Linux/PPP
hp11            IN      A       129.187.16.2
```

C.4 Zonenkonfiguration (Reverse Mapping)

```
namedb/187.129.in-addr.arpa.DNF for dfvgate.lrz-muenchen.de [129.187.10.25]
ed by ora2dns (08/94 by Heiko Hauck; Diplomarbeit TUM)

 a2824ch
 Wed Oct 26 15:43:01 MET 1994

        187.129.in-addr.arpa.
        IN      SOA     dfvgate.lrz-muenchen.de. postmaster@[...] (
                                1994082902   ; serial
                                21600        ; refresh
                                3600         ; retry
                                604800       ; expire
                                172800 )     ; minimum
        IN      NS      dfvgate.lrz-muenchen.de.
        IN      NS      hp10.lrz-muenchen.de.
        IN      NS      dfvgate.lrz-muenchen.de.
        IN      NS      hp10.lrz-muenchen.de.
        IN      PTR     dfvgate.lrz-muenchen.de.
        IN      PTR     suncollect.lrz-muenchen.de.
        IN      PTR     hp10.lrz-muenchen.de.
        IN      PTR     hp10.lrz-muenchen.de.

        IN      PTR     sun1.lrz-muenchen.de.
        IN      PTR     sun2.lrz-muenchen.de.
        IN      PTR     hp11.lrz-muenchen.de.
```

Literaturverzeichnis

[Baye 94] Rudolf Bayer, *Vorlesung Datenbanksyste*
sität München, Sommersemester 1994.

[Ecke 89] Toerless Eckert, *Unix Manual P*
Universität, Erlangen, Ja

[Frei 93] B. Freitag, *Vorles*
sit

[Geor 93] J
X.5
gar

[Golk 91] L

[RFC1010] J. Reynolds und J. Postel, „Assigned Numbers“, RFC 1010, USC Information Sciences Institute, Marina del Rey, California, Mai 1987.

[RFC1032] M. Stahl, „Domain Administrators Guide“, RFC 1032, SRI International, November 1987.

[RFC1034] P. Mockapetris, „Domain Names – Concepts and Facilities“, RFC 1034, USC Information Sciences Institute, November 1987.

[RFC1035] P. Mockapetris, „Domain Names – Implementation and Specification“, RFC 1035, USC Information Sciences Institute, November 1987.

[RFC1611] R. Austein und J. Saperia, „DNS Server MIB Extensions“, RFC 1611, Epilogue Technology Corp. / Digital Equipment Corp., Mai 1994.

[RFC1612] R. Austein und J. Saperia, „DNS Resolver MIB Extensions“, RFC 1612, Epilogue Technology Corp. / Digital Equipment Corp., Mai 1994.

[RFC881] J. Postel, „The Domain Names Plan and Schedule“, RFC 881, USC Information Sciences Institute, November 1983.

[RFC882] P. Mockapetris, „Domain names - Concepts and Facilities“, RFC 882, USC Information Sciences Institute, November 1983.

[RFC883] P. Mockapetris, „Domain names - Implementation and Specification“, RFC 883, USC Information Sciences Institute, November 1983.

[RFC974] Craig Partridge, „Mail Routing and the Domain System“, RFC 974, CSNET CIC BBN Laboratories Inc., Januar 1986.

[Sant 93] Michael Santifaller, *TCP/IP und ONC/NFS in Theorie und Praxis*, Addison-Wesley (Deutschland) GmbH, Bonn, 1993.

[Saue 92] Hermann Sauer, *Relationale Datenbanken: Theo*, Addison-Wesley (Deutschland) GmbH, Bonn,

[Teor 90] Toby J. Teorey, *Database Model*, *Relationship A*

[UCBK] University

[Valt 92]

[Valt 94] Robert Valta, *Auswertung der Umfrage zur aktuellen Netzdok*
tation, Leibniz-Rechenzentrum, Munchen, Januar 1994

[Vixi 93] Paul Vixie, *Name Server Operations Gu*
Digital Equipment Corp., Pa

[Vall 91] Larry Vall und Randal L
Associates Inc.,