

Konzeption und Implementierung einer Datenbank für Sprachsignal-Korpora

Phan Quang

Revision 5. November 1998

Inhaltsverzeichnis

1	Konzeption	5
1.1	Problemstellung	5
1.2	Aufgabenstellung	5
1.3	Methodik	5
1.4	Anforderungsanalyse und Auswahl eines Datenbanksystems	6
1.4.1	Auswahl eines Datenbanksystem	6
1.4.2	Vergleich von mSQL mit MySQL und PostgreSQL	10
1.4.3	Einige Beispiele von Datenbankoperationen	11
1.4.4	Szenario eines Datenbankzugriffs	15
1.5	Datenbankmodell und Tabellenentwurf	15
1.6	Tabellen	15
1.6.1	Datentabellen	16
1.6.2	Organisatorische Tabellen	17
1.6.3	Beziehungsdarstellende Tabellen	19
1.7	Beziehungen zwischen den Tabellen	20
1.8	Bemerkungen zum Datenbankmodell und Datenbanksystem	21
2	Programmaufbau	23
2.1	Programmmodule in C	23
2.2	Programmmodul in Perl	25
3	Ablauf des C-Programms	26
3.1	Programmstart	26
3.1.1	Partielle Automatisierung	26
3.1.2	Vollständige Automatisierung	28
3.2	Wichtige Vorbemerkungen zum Verwaltungsprogramm	29
3.2.1	Allgemeine Aspekte	29
3.2.2	Löschen von Daten aus den Tabellen	30
3.2.3	Formatumwandlungen	30
3.2.4	Datenimport in Batchform	31
3.3	Menüsystem	31

<i>INHALTSVERZEICHNIS</i>	3
3.3.1 Hauptmenüpunkt Abfragen	31
3.3.2 Untermenüpunkt Dateizuordnung löschen	37
3.3.3 Untermenü Änderungen-Prozedur	37
3.4 Funktionalitäten	38
3.4.1 Menü A	39
3.4.2 Menü A1_B	40
3.4.3 Menü A2_B	41
3.4.4 Menü A3_B	45
3.4.5 Menü A4_B	50
3.4.6 Menü A5_B	58
3.4.7 Menü A1B1_C	65
3.4.8 Menü A1B2_C	69
3.4.9 Menü A1B3_C	72
3.4.10 Menü A1B4_C	72
3.4.11 Menü A1B5_C	73
3.4.12 Menü A1B6_C	74
3.4.13 Menü A1B7_C	75
3.4.14 Menü A3B1_C	76
3.4.15 Menü A4B8_C	77
3.4.16 Menü A5B1_C	81
3.4.17 Menü A5B3_C	85
3.4.18 Menü A5B7_C	86
3.4.19 Menü A5B13_C	93
3.4.20 Menü A1B1C12_D	95
4 Perl-Schnittstelle	98
4.1 Programmstart mit Kommandozeileoptionen	98
4.2 Funktionalität	100
4.3 Aufbau der Definitionsdatei	100
4.4 Aufbau der Inputdatei	101
4.5 Aufbau der Outputdatei	102
4.6 Beispiele	102
4.6.1 Features-basierte Suche	102
4.6.2 Segment-basierte Suche	103
4.6.3 Kombinierte Suche	103
4.7 Wichtige Bemerkungen zur Perl-Schnittstelle	104
4.7.1 Allgemeine Bemerkungen	104
4.7.2 Bemerkungen zur Wertersetzung bei Kommandozeileoption %F	104
4.7.3 Bemerkungen zu Text-Dateien	105
4.8 Bemerkungen zum Perl-Tools (Hilfsroutinen)	107

5	Testphase	108
5.1	Testszenario	108
5.1.1	Bereitstellung einer leeren Sprachsignaldatenbank	108
5.1.2	Ausfüllen der Testdatenbank mit Testdaten	109
5.1.3	Löschen-Routine in Batchform testen	112
5.1.4	Formatumwandlung testen	112
5.1.5	Perl-Schnittstelle testen	113
5.1.6	Auswertung der Analyse bei der Perl-Schnittstelle	117
6	Anhang	121
6.1	Funktionsliste des C-Programms	121
6.1.1	Modul main.c	121
6.1.2	Modul hilfe1.c	122
6.1.3	Modul hilfe2.c	123
6.1.4	Modul hilfe3.c	124
6.1.5	Modul hilfe4.c	124
6.1.6	Modul hilfe5.c	125
6.1.7	Modul hilfe6.c	126
6.1.8	Modul hilfe7.c	126
6.2	Shellskripts, Perl-Tool und Dateien bei der Testphase	126
6.2.1	Shellskripts	126
6.2.2	Perl-Tools checkele.pl und checkelege.pl zur Auswertung der Analyse	127
6.2.3	Perl-Tool changepath.pl	129
6.2.4	Zusätzliche Perl-Tools	130
6.2.5	In- und Ouputdateien	130
6.3	Perl-Tools zur Erstellung der Inputdatei im C-Programm	131
6.4	API-Funktionen des mSQL-Datenbanksystems Release 1.06	133
7	Installation, Konfiguration und Verzeichnisstruktur	136
7.1	Installation des Datenbanksystems mSQL	136
7.2	Verzeichnisstruktur	137
7.3	Kompilierung und Installation vom C-Programm	138
8	Zusammenfassung und Aussicht	139
9	Literatur	141

Kapitel 1

Konzeption

1.1 Problemstellung

Am Institut für Maschinelle Sprachverarbeitung werden Sprachdaten aufgenommen sowie deren Analysen erstellt. Dabei handelt es sich überwiegend um Aufnahmen von Nachrichten und anderen Radiosendungen sowie um selbst erstellte Aufnahmen. Die Sprachsignale liegen in Form von Abtastwerten, Sprachgrundfrequenzverläufen und Audiodateien in unterschiedlichen Formaten vor. Durch automatische, halb-automatische und manuelle Segmentierung werden Text- und Labeldateien in den unterschiedlichsten Formaten erzeugt.

All diese Sprachsignale, deren Analyse und Verarbeitung ergeben Korpora, die ressourcenintensiv sind. Diese Daten sind auf Grund von äußeren Bedingungen und gewachsenen Strukturen auf verschiedenen Rechnern und Speichermedien verteilt und werden somit *von selbst* unüberschaubar und ineffektiv. Auf Grund der ständig wachsenden Datenmenge müssen diese Korpora in eine Datenbank zusammengeführt werden, die einerseits der wachsenden Datenmenge Rechnung trägt und andererseits Automatismen für einfache und schnelle Analyse und Verarbeitung bereitstellt.

1.2 Aufgabenstellung

Ziel dieser Diplomarbeit ist die Konzeption und Implementierung einer Datenbank unter Unix, welche die flexible Abfrage von Signalsegmenten in der Sprachsignal-Korpora ermöglicht.

Die Diplomarbeit soll folgende Punkte umfassen:

- 1. Anforderungsanalyse und Entwurf eines offenen Datenbankmodells. Dabei bedeutet *offenes* System die einfache Integration neuer Objekte, deren Abhängigkeiten sowie neuer Transformationsprozeduren.
- 2. Auswahl einer geeigneten Datenbankentwicklungsumgebung.
- 3. Implementierung und Dokumentation.

1.3 Methodik

Die Problem- und Aufgabenstellung lassen folgende Vorgehensweise als sinnvoll und notwendig erscheinen:

- Auswahl eines geeigneten relationalen Datenbanksystems.
- Datenbankentwurf.

- Das Verwaltungsprogramm für Sprachsignalen als Prototyp zu implementieren unter folgenden Gesichtspunkten:
 - Es soll ein handhabbares Werkzeug bereitgestellt werden, das u.a. eine Schnittstelle für Datenbankoperationen (INSERT, UPDATE, DELETE ...) zur Verfügung stellt.
 - Es soll spezifische, kontextbezogene Abfragen ermöglichen, die auf flexible Weise gestaltet und formuliert werden sollen, um auf strukturierte Informationen zugreifen zu können.
 - Darüber hinaus soll noch die Möglichkeit bestehen, Ergebnisse der Abfragen auszuwerten, abzuspeichern und evtl. an ein anderes Programm zur Weiterbearbeitung zu leiten.

1.4 Anforderungsanalyse und Auswahl eines Datenbanksystems

1.4.1 Auswahl eines Datenbanksystems

Mögliche Auswahlkandidaten

Auf Unix-Rechnern wird häufig das Datenbanksystem DBM mit installiert, das beispielsweise von dem Tool "sendmail" verwendet wird. Der Dateninhalt wird über einen Schlüsselwert dargestellt, der durch einen Struct-Type in C repräsentiert wird. Die Benutzung von DBM ist aber nicht uneingeschränkt zu empfehlen aus folgenden Gründen:

- DBM ist nicht multiuser-fähig.
- In DBM gibt es nur einen einzigen Schlüssel.
- Es gibt keine Konstrukte, um Beziehungen zwischen den Objekten direkt auszudrücken. Dies geht nur über Umwege und mit Tricks, die in einem Programm implementiert werden müssen.

Allerdings in Anbetracht einer der Anforderungen, das das im Rahmen der Diplomarbeit eingesetzte Datenbanksystem relational sein soll, können nur einige Systeme im engen Kreis betrachtet werden:

- PostgreSQL
- MySQL
- mSQL

Das Datenbanksystem mSQL und seine Vorzüge

Die Entscheidung fällt auf das Datenbanksystem mSQL (MiniSQL) aus folgenden Überlegungen, die als Pro-Argumente aufgefaßt werden können:

- Kommerzielle Datenbankprodukte sind in Anbetracht der dargestellten Problem- und Aufgabenstellung oft überdimensioniert und teuer. Als kleines, portables und preiswertes Datenbanksystem für Unix-Umgebungen hat sich das *Shareware*-Datenbanksystem mSQL im Laufe der Zeit etabliert. Dies läßt sich an einer Vielzahl von Schnittstellen zu anderen Programmiersprachen wie z.B. Perl, Tcl/Tcl und Java feststellen. Kurz und bündig gesagt stimmt das Preis-Leistungs-Verhältnis bei mSQL.

- Das Datenbanksystem `mSQL` ist eine relationale Datenbank, benutzt SQL als Datendefinitions- und Datenmanipulationssprache, bietet jedoch nicht die vollständige Funktionalität des ANSI-Standard-SQL. Für die Zielsetzung reicht aber die zur Verfügung stehende Funktionalität aus. Überdies stellt `mSQL` ein C-Programmierschnittstelle zur Verfügung, mit der Automatismen für den Zugriff auf strukturierte Informationen realisiert werden können. Die Mächtigkeit der Anbindung an `mSQL` besteht darin, daß der Programmierer ein beliebiges `mSQL`-Statement an das Datenbanksystem übergeben kann. Dabei beschränken sich die Anfragen nicht nur auf den `SELECT`-Befehl, sondern können aus jedem gültigen `mSQL`-Kommando wie `INSERT`, `UPDATE` oder `DELETE` zusammensetzen. Beziehungen zwischen den vorhandenen Objekten werden durch entsprechende Identifikatoren in den Tabellen elegant ausgedrückt.
- Zum Lieferumfang des `mSQL`-Datenbanksystems gehört eine Reihe von Datenbanktools, mit dem geübte Benutzer administrative Datenbankoperationen (Tabellen definieren, Tabellendefinition modifizieren, Tabellen löschen), datenmanipulierende Operationen (Daten in die Tabellen eintragen, Dateneinträge ändern, Dateneinträge löschen) durchführen können.
- Zusätzlich zu den oben beschriebenen Datenbanktools verfügt `mSQL` auch über eine C-Interface, mit deren Hilfe Programme erstellt werden können, die auf Daten einer `mSQL`-Datenbank zugreifen. Dies geschieht selbstverständlich auf SQL-Niveau. D.h., eine SQL-Anweisung wird in einen String gepackt und mittels einer geeigneten C-API-Funktion an das `mSQL`-Datenbanksystem übermittelt, das die Ergebnismenge zurückliefert.
- Wegen seiner Größe und seiner Geschwindigkeit ist `mSQL` für die obengenannte Aufgabenstellung einsetzbar.
- `mSQL` unterstützt auch die Anbindung zum Datenbankserver über das gängige Netzwerkprotokoll TCP/IP in der Unix-Umgebung. Somit wird der Fall abgedeckt, daß die Sprachsignalen über Rechnergrenzen hinweg verteilt sind.
- `mSQL` besitzt außerdem eine Schnittstelle für Web-Anwendungen. Auf diese Weise lassen sich SQL-Kommandos mit geringem Programmieraufwand in HTML-Seiten einbetten. Diese Schnittstelle wird als `W3-mSQL` benannt und ermöglicht die Anbindung seines Datenbanksystems an beliebige HTTP-Server. Diese Möglichkeit erweist sich als günstig für den Fall, daß Zugriffe auf Informationen über das WWW-Protokoll in Zukunft erfolgen soll. Zu diesem Zweck existiert bereits eine spezielle API-Schnittstelle für die gebräuchliche Programmiersprache Perl.
- Die Zugriffskontrolle wird über eine Kontrolldatei mit der Endung `acl` geregelt, in die entsprechende Einträge gemacht werden. Somit ist eine gesicherte Zugriffskontrolle ausreichend vorgesorgt. Jede Änderung an der Kontrolldatei muß aktiviert werden. Dies kann durch den Befehl `mSQL reload` zur Laufzeit bewerkstelligt werden.
- Es ist unter `mSQL` möglich, einen **Dump** der Datenbank im ASCII-Format zu erstellen. Dieser Dump enthält alle notwendigen SQL-Befehle zur Rekonstruktion der Datenbank. Dadurch ist es möglich, die Datenbankinhalte in eine andere SQL-Datenbank zu portieren.
- Mit wenigen Handgriffen läßt sich `mSQL` mit Hilfe einer mitgelieferten Makefile-Datei leicht compilieren und installieren, gesetzt den Fall, daß man entsprechende Rechte auf einem Unix-Rechner besitzt. In der im gesamten Softwarepaket mitgelieferten Datei `README` wird verständlich beschrieben, wie die Installation des `mSQL`-Datenbanksystems vorgenommen werden kann (siehe 7.2). Die Angabe eines Verzeichnisses für die `PID`-Datei setzt voraus, daß der Benutzer ebenfalls Schreibrecht für dieses Verzeichnis besitzt.

Einschränkungen von `mSQL`

Der Name des verwendeten Datenbanksystems *MiniSQL* suggeriert bereits einige Einschränkungen in Bezug auf die gesamten Funktionalitäten.

Die erste Einschränkung betrifft die in der Diplomarbeit eingesetzte Release 1.06 auf Grund der Entwicklungsplattform SGI. Diese Release ist nicht die aktuellste, denn die neueste Release mit ihren umfangreicheren Funktionalitäten läßt sich auf der SGI-Plattform leider nicht compilieren und installieren. Beispielsweise werden nur folgende Datentypen bei dieser Version unterstützt,

- int (Integerzahl)
- char (für Zeichenketten)
- real (für reelle Zahlen)

so daß man bei der Erstellung der Tabellenfelder nicht so viel Spielraum hat für die Formatgestaltung der Tabellenfelder. In der verwendeten mSQL-Version wird das Indizieren von Tabellenfeldern auch nicht unterstützt, das in der Regel dient, schnellere Zugriffe auf die Daten zu ermöglichen. Außerdem fehlen mehrere Standard-Mechanismen, die bei relationalen Datenbanksystemen in der Regel implementiert sind, wie z.B. VIEWS, STORED PROCEDURES, VALIDATIONSREGEL bezogen auf Tabellenspalten Der Zugriff auf eine mSQL-Datenbank wird ausschließlich über die Acl-Kontrolldatei geregelt. Es ist also nicht möglich, Zugriffsrechte bezogen auf SQL-Statement und/oder auf Tabellenspalten zu definieren, weil das mSQL-Datenbanksystem diese Möglichkeit nicht anbietet.

Folgende Auflistung gibt einen kurzen Überblick über alle unterstützten SQL-Befehle bei der vorliegenden Version, die ja nur eine Teilmenge des ANSI-SQL-Sprachumfangs (Standard-Version) sind:

- Tabelle erzeugen:

```
CREATE TABLE table_name (
  col_name col_type [ not null | primary key ]
  [, col_name col_type [ not null | primary key ] ] ** )
```

- Tabellen löschen:

```
DROP TABLE table_name
```

- Werte in die Tabellen eintragen:

```
INSERT INTO table_name [ ( column [ , column ] ** ) ]
VALUES ( values [, value ] ** )
```

- Werte in den Tabellen löschen:

```
DELETE FROM table_name
WHERE column OPERATOR value
[ AND | OR column OPERATOR value ] **
OPERATOR kann folgende Werte annehmen: < , > , = , <= , >= , <> , LIKE
```

- Abfrage formulieren:

```

SELECT [ DISTINCT ] [table.]column [ , [table.]column ] **
FROM table [ = alias ] [ , table [ = alias ] ] **
[ WHERE [table.]column OPERATOR VALUE
[ AND | OR [table.]column OPERATOR VALUE ] ** ]
[ ORDER BY [table.]column [DESC] [ , [table.]column [DESC] ]
[limit n ]
VALUE ist entweder ein Spaltenname oder ein Wert.

```

- Werte in den Tabellen ändern:

```

UPDATE table_name SET column=value [ , column=value ]**
WHERE column OPERATOR value
[ AND | OR column OPERATOR value ]**

```

Diese Einschränkung ist nicht sehr tragisch, entsteht nur im Zusammenhang mit der Entwicklungsumgebung und stellt daher keinen "hausgemachten" Nachteil im übertragenen Sinne dar. Für die Implementierung des Prototyps kann man mit den drei unterstützten Formaten **int**, **char** und **real** auskommen.

Die zweite Einschränkung geht mit den gesamten Datenvolumen einher. Bezüglich des Speicherplatzes kann jede Tabelle theoretisch maximal 4GB groß sein. Die Zugriffsgeschwindigkeit und -zuverlässigkeit leidet allerdings runter, wenn diese kritische Größe angenähert wird.

In kommerziellen Datenbanksystemen gibt es Mechanismen, die die referentielle Integrität bei datenmanipulierenden Operationen (DML-Operationen wie INSERT, DELETE, UPDATE überwachen. Beispiele dafür sind wie z.B. Trigger, Constraint-Regel und deklarative Datenintegrität (z.B. Propagation beim Löschen eines Datensatzes). Bei mSQL muß zum einen der Datenbankadministrator durch gezielte Operationen oder das zu implementierende Programm durch entsprechende Implementierung und im Rahmen der Möglichkeiten selber dafür sorgen, daß die Daten konsistent bleiben. Zum anderen muß das zu implementierende Verwaltungsprogramm dafür sorgen, daß die Datenintegrität nicht verletzt wird. Beispielsweise soll sichergestellt werden, daß die ID-Nummer jenes Dateneintrags eindeutig ist, indem die neu zu vergebende ID-Nummer bei jedem INSERT-Vorgang errechnet wird. Dieser Mechanismus wird in der Datenbankliteratur als **programmatische Integrität** bezeichnet. Die verwendete Version 1.06 des mSQL-Datenbanksystems unterstützt zumal die Indizierung der Daten nicht. Mit Hilfe von Indizes kann man bekanntlich unter anderem den Suchvorgang bei Anfragen beschleunigen. Aus diesem Grund wird die Performance des Datenbanksystems immer schlechter, wenn sich Datenbestände im Laufe der Zeit ansammeln.

Darüber hinaus unterstützt das Datenbanksystem mSQL keine binäre Speicherstruktur wie z.B. BLOB, so daß beispielsweise der physische Inhalt einer Sd-Datei nicht direkt in der Datenbank untergebracht werden kann. Es ist nur möglich, den Namen einer Sprachsignaldatei in Textform in der Datenbank zu speichern. Auf diese Weise enthält die Datenbank **Metadaten** über Sprachsignaldateien. Dies kann aber als Vorteil betrachtet werden, weil dadurch die eigentliche Datenbank bezogen auf seine Kapazität und den damit verbundenen benötigten Speicherplatz nicht so groß wird.

Administrativer Funktionsumfang von mSQL

Trotz nicht zu übersehenden Einschränkungen kann das Datenbanksystem mSQL im Rahmen der Aufgabenstellung eingesetzt werden. Zusammen mit der C-Schnittstelle gehören einige lauffähige Programm-Tools für die Datenbankadministration zum Lieferumfang von mSQL:

mysqladmin: Datenbank kreieren, Datenbank löschen, Datenbankserver herunterfahren.

Beispiele:

mysqladmin create signal: leere Datenbank "signal" wird erzeugt.

mysqladmin drop signal: Datenbank "signal" wird gelöscht.

mysqladmin shutdown: Datenbankserver auf dem lokalen Rechner wird runtergefahren. Dieser Aufruf ist manchmal erforderlich, falls man den Datenbankserver nicht mehr starten kann. In diesem Fall soll zuerst "mysqladmin shutdown" eingegeben werden, bevor der Datenbankserver mit "msqld" gestartet wird.

mysqldump: Gesamtinhalt der Datenbank in eine Textdatei zu transportieren oder eine Backup-Version von der Datenbank erzeugen,

mysqldump signal > signal.dmp

oder Textdatei in die Datenbank wieder rücktransportieren.

mysqldump signal < signal.dmp

Auf diese Weise kann man sehr schnell ein Backup von der gesamten Datenbank machen und sichern.

mysql: Mit diesem sogenannten Monitor kann man u.a. manuell Tabellen erzeugen, löschen, Datensätze ändern per SQL-Anweisungen.

Am Anfang der Datenbankmodellierung wird dieses Programm verwendet, um Tabellen zu erzeugen und Datensätze einzutragen. Danach kann man von einem zu implementierenden Programm via C-API Datenoperationen durchführen.

relshow: Dieses Programm zeigt die Struktur der Datenbank und der darin enthaltenen Tabellen an.

Beispiel:

relshow signal (zeigt die Struktur der Datenbank "signal")

relshow signal file (zeigt die Struktur der Tabelle "FILE" in der Datenbank "signal")

msqld: Dieses Programm fungiert als Datenbankserver-Dämon und muß unbedingt vor jeder Sitzung oder beim Reboot gestartet werden.

1.4.2 Vergleich von mSQL mit MySQL und PostgreSQL

Das relationale Datenbanksystem MySQL ist mSQL im Hinblick auf seine Architektur, seine Funktionalität und seine Arbeitsweise am nächsten. MySQL ist mSQL in fast allen Belangen überlegen, es hat mehr eingebaute Datentypen (Datum, Zeitstempel, BLOB ...), unterstützt die Indizierung von Daten, stellt ebenfalls ein C-API dem Programmierer zur Verfügung. Kurz um: MySQL ist der größere Bruder von mSQL. Für die Entwicklungsumgebung SGI IRIX müssen aber mehrere Patches installiert werden. Überdies gibt es noch ein kritisches Problem: IRIX 6.2 unterstützt POSIX Threads nicht, was bei MySQL erforderlich ist, so daß die Installation wohl Schwierigkeiten bereiten wird.

Im Vergleich zu mSQL und MySQL ist das Datenbanksystem PostgreSQL hinsichtlich der zur Verfügung stehenden Funktionalitäten sehr viel mächtiger, verfolgt konsequenter das Client-Server-Paradigma, nicht so leicht zu installieren wie bei mSQL und MySQL und eher objektorientiert (eine Tabelle wird als eine Klasse betrachtet).

Aus der Gegenüberstellung von den drei genannten relationalen Datenbanksystemen kristallisieren sich folgenden Feststellungen heraus, die letztendlich entscheidend sind zugunsten von mSQL:

- Im Rahmen der Diplomarbeit und auch zeitlich gesehen ist es nicht gerechtfertigt, soviel Zeit und Aufwand in die Installation von einem umfangreichen Datenbanksystem wie PostgreSQL zu investieren.
- An Stelle von mSQL kann man auch MySQL verwenden. Die Installation von MySQL gestaltet sich uner IRIX jedoch kompliziert und kann bestenfalls von einem Systemadministrator mit ausreichenden Schreibrechten vorgenommen werden.

1.4.3 Einige Beispiele von Datenbankoperationen

In diesem Abschnitt werden einige wichtige Datenbankoperationen erläutert, die in erster Linie für die problemlose Arbeit mit der Datenbank von Hilfe sind:

Erstellung einer neuen Datenbank ohne Daten

Es gibt grundsätzlich mehrere Möglichkeiten dafür, die sich nur geringfügig unterscheiden. Folgende Methode wäre die geeigneteste:

- Eine Dump-Datei der bestehenden Signal-Datenbank erzeugen mit dem Tool msqldump:
 - Aufruf: `msqldump "Datenbankname" > "Name der Dump – Datei"`
 - Beispiel: `msqldump signal > signal.dump`
- Die Dump-Datei entsprechend modifizieren, indem die Dateneinträge von den Tabellen aus der Dump-Datei gelöscht werden:

Beispiel: Auszug von der Datei signal.dump

Dieser Teil bleibt

```
#
# Table structure for table 'file'
#
CREATE TABLE file (
  ID INT NOT NULL PRIMARY KEY,
  name CHAR(256) NOT NULL,
  pre CHAR(256),
  datum CHAR(10),
  uhrzeit CHAR(8),
  endung CHAR(8) NOT NULL,
  pfad CHAR(256) NOT NULL,
  quelle CHAR(256)
) \g
```

```
--> -----
```

```
--> -----
```

Dieser Teil wird gel"oscht

```
#
# Dumping data for table 'file'
```

#

```

INSERT INTO file VALUES
(113,'dlf970430.1656.words','dlf','30.04.97','','','words','/graugans/cd/Sternzeit3/
/April\_97/','SZ3\_April97')\g
INSERT INTO file VALUES
(112,'dlf970430.1656.txt','dlf','30.04.97','','','txt','/graugans/cd/Sternzeit3/Apr
il\_97/','SZ3\_April97')\g
INSERT INTO file VALUES
(111,'dlf970430.1656.sd','dlf','30.04.97','','','sd','/graugans/cd/Sternzeit3/April
\_97/','SZ3\_April97')\g
INSERT INTO file VALUES
(110,'dlf970430.1656.mfc','dlf','30.04.97','','','mfc','/graugans/cd/Sternzeit3/Apr
il\_97/','SZ3\_April97')\g
INSERT INTO file VALUES
(109,'dlf970429.1656.words','dlf','29.04.97','','','words','/graugans/cd/Sternzeit3/
/April\_97/','SZ3\_April97')\g
INSERT INTO file VALUES
(108,'dlf970429.1656.txt','dlf','29.04.97','','','txt','/graugans/cd/Sternzeit3/Apr
il\_97/','SZ3\_April97')\g
INSERT INTO file VALUES
(107,'dlf970429.1656.sd','dlf','29.04.97','','','sd','/graugans/cd/Sternzeit3/April
\_97/','SZ3\_April97')\g
INSERT INTO file VALUES
(106,'dlf970429.1656.mfc','dlf','29.04.97','','','mfc','/graugans/cd/Sternzeit3/Apr
il\_97/','SZ3\_April97')\g
INSERT INTO file VALUES
(105,'dlf970426.1656.words','dlf','26.04.97','','','words','/graugans/cd/Sternzeit3/
/April\_97/','SZ3\_April97')\g
--> -----

```

Dieser Vorgang wird für jede Tabelle in der Datenbank durchgeführt. Am Schluss enthält die betroffene Dump-Datei nur die Definition der Tabellen. Alle Dateinträge (INSERT) werden bereinigt. Es muß hier darauf hingewiesen werden, daß jede SQL-Anweisung mit dem Befehl "\g" abgeschlossen werden muß.

- In diesem Schritt wird eine neue Datenbank mit Hilfe des Tools `msqladmin` erzeugt.
 - Aufruf: `msqladmin create "Datenbankname"`
 - Beispiel: **`msqladmin create test`**
- Nach obigem Schritt erhält man eine neue Datenbank namens "test" ohne Tabellen. Es gilt nun in diesem Schritt, Tabellen in der Datenbank "test" zu erzeugen und sie mit Daten auszufüllen. Dazu gibt es zwei Möglichkeiten:
 - 1. Möglichkeit: interaktiv
Mit dem Monitor-Programm `msql` kann man interaktiv SQL-Befehl ausführen. Dies geht z.B. mit folgenden Befehlssequenzen:

Tabelle file erzeugen:

```

- msql test<RETURN>
- CREATE TABLE file (
  ID INT NOT NULL PRIMARY KEY,
  name CHAR(256) NOT NULL,
  pre CHAR(256),
  datum CHAR(10),
  uhrzeit CHAR(8),
  endung CHAR(8) NOT NULL,
  pfad CHAR(256) NOT NULL,
  quelle CHAR(256)
) \g<RETURN>

```

- 2. Möglichkeit: mittels einer Definitionsdatei

Die Definitionsdatei enthält in diesem Fall alle SQL-Statements (CREATE) zum Erzeugen der Tabellen. Nun wird die vorher geänderte Dump-Datei als Definitionsdatei herangezogen. Auch hier bedient man sich der Hilfe des Monitor-Programms msql:

- * Aufruf: *msql "Datenbankname" < "Definitionsdatei"*
- * Beispiel: *msql test < signal.dump*

- Auf diese Weise werden auf einmal alle Tabellen kreiert.

Eintragen von Daten in eine leere Datenbank

Als Beispiel sollen Daten in die Tabelle FILE hinzugefügt werden. Dafür gibt es insgesamt drei Methoden:

- 1. Interaktiv:

- Das Monitor-Programm msql wird gestartet. Danach werden INSERT-Statements eingegeben.

Beispiel: *msql test < RETURN >*

```

INSERT INTO file VALUES (113,'dlf970430.1656.words','dlf','30.04.97','','
'words','/graugans/cd/Sternzeit3/April\_97/','SZ3\_April97')
\g<RETURN>

```

oder

Das C-Programm wird gestartet, dann wird der Menüpunkt "Datenbankroutinen-Eintragen" ausgewählt. Danach wird der Menüpunkt "Datei eintragen" angesteuert und anschließend der Menüpunkt "interaktiv" ausgewählt

- 2. Mittels einer Definitionsdatei: Dies geschieht in folgenden Schritten:

- Es wird eine Dump-Datei test.dump von der leeren Datenbank test erzeugt:
msqldump test > test.dump
- In die Dump-Datei "test.dump" werden INSERT-Statements mit einem gewöhnlichen Editor hinzugefügt.
- Mit dem Befehl *msql test < test.dump* werden Daten in die Datenbank übertragen

- 3. Das Programm bietet einen Menüpunkt an zum Einlesen von Daten, die in einer Inputdatei untergebracht sind. Diese Inputdatei muß aber ein bestimmtes, vordefiniertes Format aufweisen. Im folgenden wird zur Illustration ein kurzer Auszug aus der Datei words.in angezeigt:

```
#! PATH NAME PRE DATE TIME EXT

/graugans/cd/stern1/words/ dlf960724.1657.words dlf 24.07.96 1657 words
/graugans/cd/stern1/words/ dlf960731.1657.words dlf 31.07.96 1657 words
/graugans/cd/stern1/words/ dlf960801.1656.words dlf 01.08.96 1656 words
/graugans/cd/stern1/words/ dlf960802.1656.words dlf 02.08.96 1656 words
/graugans/cd/stern1/words/ dlf960803.1656.words dlf 03.08.96 1656 words
/graugans/cd/stern1/words/ dlf960804.1656.words dlf 04.08.96 1656 words
```

Es muß an dieser Stelle eindringlich hervorgehoben werden, daß die Konfigurationszeile **#! PATH NAME PRE DATE TIME EXT** unbedingt anzugeben ist, damit die Einlese- oder die Löschroutine mit der Inputdatei anfangen kann. Es können nur die Elemente PATH, NAME, PRE, DATE, TIME und EXT spezifiziert werden. Die Gesamtanzahl der spezifizierten Elemente beträgt maximal 6. Es können aber weniger als 6 Elemente in der Konfigurationszeile angegeben werden. Überdies muß besonder darauf geachtet werden, daß zumindest die drei Elemente PATH, NAME und EXT bei der Einlese-Option angegeben werden müssen. Es ist hierbei anzumerken, daß dies in der implementierten Einlese-Routine auf jeden Fall überprüft wird. Die Reihenfolge der Elemente innerhalb dieser Konfigurationszeile ist aber beliebig austauschbar. Folgende Beispiele zeigen korrekte Konfigurationen:

- #! PATH NAME PRE DATE TIME EXT
- #! NAME PRE PATH TIME EXT DATE
- #! NAME PATH TIME EXT

Die Konfigurationszeile ermöglicht die Interpretation der Daten, die in durch Leer- oder Tabulatorzeichen getrennten Spalten angeordnet sind. Der Benutzer muß auf jeden Fall dafür sorgen, daß die Daten mit der Konfiguration konform sind.

Beispiel einer Kontrolldatei für Datenbankzugriffe

```
#Zugriffskontrolle für mSQL
database=signal
read=bambi,paul
write=root
host = *.ims.uni-stuttgart.de,-sperber.ims.uni-stuttgart.de
access=local,remote
```

Der Aufbau der Kontrolldatei besteht aus mehreren Einträgen, wie das obige Exemplar zeigt. Dabei wird das Zugriffsrecht wie folgt geregelt: Die Datenbank "signal" kann von allen Hosts innerhalb der Domäne ims.uni-stuttgart.de angebunden werden, jedoch mit Ausnahme vom Host sperber.ims.uni-stuttgart.de. Die Anbindung zur Datenbank kann sowohl über einen lokalen als auch über einen Remote-Datenbankserver abgewickelt werden. Das Leserecht erhalten im Beispiel die Benutzer *bambi* und *paul*. Das Schreibrecht hat nur der Benutzer *root*, der das Datenbanksystem mSQL installiert hat. Es gilt zu beachten, daß bei der Festlegung der Zugriffsrechte (Schreiben, Lesen) auch der Platzhalter "*" eingesetzt werden kann. Der Zugriffsentzug wird durch ein vorangestelltes Minus-Zeichen erkennbar gemacht.

1.4.4 Szenario eines Datenbankzugriffs

Die C-Schnittstelle ist leistungsfähig genug, um mit deren Hilfe von einem C-Programm aus auf die Daten zugreifen zu können. Der Ablauf eines Zugriffs kann wie folgt skizziert werden:

- Per `mysqlconnect()` wird eine Verbindung zum Datenbankserver hergestellt. Diese Funktion liefert einen Handle zurück (eine Integerzahl), die bei weiteren Zugriffen unbedingt angegeben werden muß, sofern kein Fehler bei der Verbindung festgestellt wird. Diese Operation braucht im Programm nur einmal ausgeführt zu werden, es sei denn, man braucht gleichzeitig verschiedene Verbindungsmöglichkeiten zum Datenbankserver. In diesem Fall wird `mysqlconnect()` mehrmals gestartet. Jedes mal wird eine andere Integerzahl zurückgeliefert. `mysqlconnect()` unterstützt auch die Remote-Anbindung über Netz durch Angabe entsprechender Parameter.
 - Wenn **NULL** as Parameter angegeben wird, dann wird versucht, eine Verbindung zum lokalen Datenbankserver aufzubauen.
 - Andernfalls muß entweder ein Hostname oder eine IP-Adresse als Parameter angegeben werden. Entsprechend ist dieser Remote-Datenbankserver für weitere Datenbankzugriffe verantwortlich.
- Als nächstes wird eine Datenbank mittels der Funktion `mysqlSelectDB()` ausgewählt. Als Parameter ist der Name der Datenbank anzugeben, der per Kommandozeileoption (-s DB-Name) bekannt gemacht werden kann.
- Nun wird eine SQL-Anweisung formuliert, die zusammen mit dem Handle der Funktion `mysqlQueryDB()` als Parameter übergeben werden. Danach wird `mysqlQueryDB()` ausgeführt. Das Ergebnis dieser Operation wird in der vordefinierten Datenstruktur `m_result` (definiert in der Headerdatei `mysql.h`) durch Aufruf der Funktion `mysqlStoreResult()` abgespeichert. Es wird dabei vorausgesetzt, daß die SQL-Anweisung syntaktisch und semantisch richtig erstellt wird.
- Je nach Bedarf kann man anhand der Funktion `mysqlNumRows` die Gesamtanzahl der Daten ermitteln. Daraufbauend kann eine Durchlaufschleife implementiert werden, innerhalb derer sukzessiv auf die Daten zugegriffen wird. Der Zugriff auf die Daten erfolgt bei jedem Durchlauf durch die Funktion `mysqlFetchRow()`, die Daten aus der Tabelle zeilenweise in Form einer Feld-Struktur zurückliefert, auf deren einzelne Positionswerte zugegriffen werden kann.

1.5 Datenbankmodell und Tabellenentwurf

Unter Datenbankmodell versteht man in erster Linie das Datenbankschema, das im wesentlichen aus Tabellen und deren wechselseitigen Beziehungen besteht. Die Tabellenstruktur soll relevante, möglichst wenig redundante Informationen sowohl unter organisatorischem als auch unter programmtechnischem Aspekt beinhalten, die allerdings alle notwendigen zwecksgebundenen Abfragemöglichkeiten entsprechend der applikationsbedingten Aufgabenstellung abdecken sollen. Darüber hinaus soll das Datenbankmodell so angelegt werden, daß es mögliche Erweiterungen im Aufbau ermöglicht. Im folgenden werden im ersten Schritt die verwendeten Tabellen aufgelistet und deren Bedeutungen erläutert. Im nächsten Schritt werden ihre Beziehungen zueinander in Betracht gezogen.

1.6 Tabellen

Im jetzigen Stand gibt es insgesamt 20 Tabellen. Diese Tabellenstruktur orientiert sich in erster Linie an der Aufgabenstellung und an der Beschaffenheit der zu verwaltenden Sprachsignalen. Beispielsweise

enthält die Tabelle FILE die Spalten PRE, DATUM, ENDUNG, weil der gesamte Dateiname einer Sprachsignaldatei solche Informationen enthält. Unter programmtechnischem Gesichtspunkt werden die Tabellen in drei Kategorien untergliedert:

- Datentabellen
- Organisatorische Tabellen
- Beziehungsdarstellende Tabellen (Zuordnungstabellen)

1.6.1 Datentabellen

Tabellen dieser Kategorie enthalten relevante applikationsbedingte Informationen, die unmittelbar dem Applikationsbereich entstammen und in gewisser Hinsicht die Realität modellieren.

Tabelle FILE Diese Tabelle ist die wichtigste überhaupt in der gesamten Datenbank, denn sie enthält relevante Informationen über Sprachsignaldateien. Sie hat insgesamt 8 Felder:

Feldname	Typ	Bedeutung	Leer
ID	int	Identifikator, Primärschlüssel	Nein
name	char(256)	Name der Signaldatei	Nein
pre	char(256)	Dateistamm	Ja
datum	char(256)	Datum der Signaldatei	Ja
uhrzeit	char(256)	Uhrzeit, wann die Signaldatei erzeugt wird	Ja
endung	char(256)	Dateiextension	Nein
pfad	char(256)	Pfad, wo die Signaldatei zu finden ist	Nein
quelle	char(256)	Quelle, von der die Signaldatei stammt	Ja

Tabelle PROZEDUR Diese Tabelle enthält Daten über Programme, die z.B. für die Umwandlung der Signale in ein anderes Format herangezogen werden.

Feldname	Typ	Bedeutung	Leer
ID	int	Identifikator, Primärschlüssel	Nein
name	char(256)	Name der Prozedur	Nein
typ	char(256)	Prozedurtyp	Ja

Tabelle FORMAT In dieser Tabelle werden die in Frage kommenden Formate der Sprachsignaldateien gespeichert. Es ist offensichtlich, daß man anhand der Dateiextension das jeweilige Format eines Datentyps extrahieren kann. Aus organisatorischen und programmtechnischen Gründen werden die Informationen über Formate jedoch separat in dieser Tabelle untergebracht. Dies ermöglicht einen schnelleren Zugriff auf die erwünschten Daten und macht den Zugriff übersichtlicher.

Feldname	Typ	Bedeutung	Leer
ID	int	Identifikator, Primärschlüssel	Nein
kennzeichnung	char(256)	Kennzeichnung des Formats	Nein

Tabelle HAUPTTYP Jedes Signal besitzt in der Regel einen Haupttyp. Diese Tabelle wird erstellt, um die vorkommenden Haupttypen unterzubringen.

Feldname	Typ	Bedeutung	Leer
ID	int	Identifikator, Primärschlüssel	Nein
kennzeichnung	char(256)	Kennzeichnung des Haupttyps	Nein

Tabelle UNTERTYP Die Typen der Signale können weiter in Untertypen aufgeteilt werden. Diesem Sachverhalt trägt diese Tabelle Rechnung.

Feldname	Typ	Bedeutung	Leer
ID	int	Identifikator, Primärschlüssel	Nein
kennzeichnung	char(256)	Kennzeichnung des Untertyps	Nein

Tabelle QUELLE Die Signale stammen aus einigen Quellen (Dateien). In dieser Tabelle werden Information über die Quelle festgehalten.

Feldname	Typ	Bedeutung	Leer
ID	int	Identifikator, Primärschlüssel	Nein
name	char(256)	Name der Quelle	Nein
name2	char(256)	weiterer Name der Quelle	Ja

Tabelle ZUGRIFFSRECHT Im Gegensatz zu anderen vorhergenannten Tabellen ist die Struktur der Tabelle Zugriffsrecht noch nicht festgelegt. Diese Tabelle wird verwendet, um Benutzerzugriffe auf Signaldateien wie auf UNIX-Ebene abzubilden. Das Feld "wie" kann die Werte + oder - annehmen, wobei + Zugriffserteilung und - Zugriffsentzug bedeutet.

Feldname	Typ	Bedeutung	Leer
ID	int	Identifikator, Primärschlüssel	Nein
wie	char(2)	mögliche Werte +/-	Nein
was	char(10)	Kombination aus r,w,x	Nein

1.6.2 Organisatorische Tabellen

Tabellen dieser Kategorie haben in erster Linie nicht direkt mit der Applikation zu tun. Sie sind in organisatorischer Hinsicht Hilfstabellen und stellen deshalb Daten zur Verfügung, die ein reibungsloses Zusammenspiel im Aufbau der Datenbank, im Aufrechterhalten der referentiellen Integrität zwischen den Tabellen ermöglicht. Alle folgenden Tabellen sorgen jeweils beispielsweise dafür, daß der Identifikator (also der Primärschlüssel) jedes Datensatzes in der entsprechenden Tabelle eindeutig ist, da das verwendete Datenbanksystem keinen direkten Mechanismus zur Ermittlung eines neuen, eindeutigen Werts zur Verfügung stellt (im Sinne eines selbst inkrementierenden Autowerts).

Tabelle FILE_ID Jede Sprachsignaldatei in der Tabelle FILE wird durch eine eindeutige Integernummer als Identifikator identifiziert. Beispielsweise ist beim Eintragen einer neuen Sprachsignaldatei in diese Tabelle ein neuer noch nicht verwendeter Identifikator erforderlich. Diese Tabelle enthält die aktuelle verwendete Nummer, die nächste zu verwendende Nummer ist die aktuelle Nummer plus 1.

Feldname	Typ	Bedeutung	Leer
ID	int	aktuelle Identifikatornummer, verwendet bei FILE	Nein

Tabelle PROZEDUR_ID Jede Prozedur in der Tabelle PROZEDUR wird durch eine eindeutige Integernummer als Identifikator identifiziert. Die Tabelle PROZEDUR_ID enthält die aktuell besetzte Nummer als Identifikator.

Feldname	Typ	Bedeutung	Leer
ID	int	aktuelle Identifikatornummer, verwendet bei PROZEDUR	Nein

Tabelle FORMAT_ID Jedes Format in der Tabelle FORMAT wird durch eine eindeutige Integernummer als Identifikator identifiziert. Die Tabelle FORMAT_ID enthält die aktuell besetzte Nummer als Identifikator.

Feldname	Typ	Bedeutung	Leer
ID	int	aktuelle Identifikatornummer, verwendet bei FORMAT	Nein

Tabelle HAUPTTYP_ID Jeder Haupttyp in der Tabelle HAUPTTYP wird durch eine eindeutige Integernummer als Identifikator identifiziert. Die Tabelle HAUPTTYP_ID enthält die aktuell besetzte Nummer als Identifikator.

Feldname	Typ	Bedeutung	Leer
ID	int	aktuelle Identifikatornummer, verwendet bey HAUPTTYP	Nein

Tabelle UNTERTYP_ID Jeder Untertyp in der Tabelle UNTERTYP wird durch eine eindeutige Integernummer als Identifikator identifiziert. Die Tabelle UNTERTYP_ID enthält die aktuell besetzte Nummer als Identifikator.

Feldname	Typ	Bedeutung	Leer
ID	int	aktuelle Identifikatornummer, verwendet bei UNTERTYP	Nein

Tabelle QUELLE_ID Jede Datenquelle in der Tabelle QUELLE wird durch eine eindeutige Integernummer als Identifikator identifiziert. Die Tabelle QUELLE_ID enthält die aktuell besetzte Nummer als Identifikator.

Feldname	Typ	Bedeutung	Leer
ID	int	aktuelle Identifikatornummer, verwendet bei QUELLE	Nein

Tabelle FILE_SPH Diese Tabelle kann als eine temporäre Tabelle betrachtet werden. Sie enthält Dateinamen, die Sph-Format haben. Bei der Formatumwandlung nach Sph-Format (Menüpunkt A=2) wird jede umgewandelte Datei in diese Tabelle eingetragen. Somit dient die Tabelle FILE_SPH u.a. der Überprüfung, ob eine Datei überhaupt umgewandelt wurde oder noch nicht.

Feldname	Typ	Bedeutung	Leer
ID	int	aktuelle Identifikatornummer	Nein
hline name	char	Dateiname	Nein
file_quelle	char	Quelle der Datei	Ja
pfad	char	vollständiger Dateipfad	Nein

Tabelle FILE_PHONES Die Tabelle FILE_PHONES erfüllt denselben Zweck wie die Tabelle FILE_SPH und wird verwendet bei der Formatumwandlung nach Phones-Format.

Feldname	Typ	Bedeutung	Leer
ID	int	aktuelle Identifikatornummer	Nein
hline name	char	Dateiname	Nein
file_quelle	char	Quelle der Datei	Ja
pfad	char	vollständiger Dateipfad	Nein

Tabelle FILE_WORDS Die Tabelle FILE_WORDS erfüllt denselben Zweck wie die Tabelle FILE_SPH und wird verwendet bei der Formatumwandlung nach Words-Format.

Feldname	Typ	Bedeutung	Leer
ID	int	aktuelle Identifikatornummer	Nein
hline name	char	Dateiname	Nein
file_quelle	char	Quelle der Datei	Ja
pfad	char	vollständiger Dateipfad	Nein

Tabelle FILE_TXT Die Tabelle FILE_TXT erfüllt denselben Zweck wie die Tabelle FILE_SPH und wird verwendet bei der Formatumwandlung nach Text-Format.

Feldname	Typ	Bedeutung	Leer
ID	int	aktuelle Identifikatornummer	Nein
hline name	char	Dateiname	Nein
file_quelle	char	Quelle der Datei	Ja
pfad	char	vollständiger Dateipfad	Nein

1.6.3 Beziehungsdarstellende Tabellen

Tabellen dieser Kategorie drücken im Prinzip die wechselseitigen Beziehungen zueinander aus. Somit wird unter anderem eine Assoziationsordnung zwischen der Haupttabelle FILE und anderen Tabellen realisiert, welche die Eigenschaften der Sprachsignaldateien darstellen.

Tabelle FILEEXT_PROZEDUR_ZU Diese Tabelle enthält Informationen über die Beziehung zwischen einer Sprachsignaldateiextension und einer Prozedur. Jeder Datensatz der Tabelle verknüpft somit eine Dateiextension mit einer Prozedur. Auf diese Weise kann man bei der Formatumwandlung von einem Format zum andern herausfinden, welche Prozedur zu diesem Zweck notwendig ist in Abhängigkeit von einer vorliegenden Dateiextension.

Diese Tabelle enthält auch die Dateiextension der Ergebnisdatei einer Prozedur, falls diese Prozedur auf eine Sprachsignaldatei angewandt wird. Die Information über die entsprechende Dateiextension der Ergebnisdatei wird u.a. intensiv verwendet bei Formatumwandlungsroutinen.

Feldname	Typ	Bedeutung	Leer
ext	char(256)	Dateiextension	Nein
pro_name	char(256)	verknüpfter Prozedurname	Nein
outext	char(256)	Dateiextension der Zieldatei	Nein

Tabelle FILE_PRO_ZUORDNUNG Diese Tabelle erfüllt den gleichen Zweck wie bei der Tabelle FILEEXT_PROZEDUR_ZU, enthält daher redundante Informationen. Jeder Datensatz dieser Tabelle stellt eine Beziehung zwischen einer Signaldatei (vertreten durch den entsprechenden Identifikator) und einer Prozedur (auch vertreten durch einen Identifikator) dar. Es ist möglich, daß im Laufe der Programmentwicklung diese Tabelle eliminiert wird.

Feldname	Typ	Bedeutung	Leer
ID	int	Identifikator der Signaldatei	Nein
FLP_ZU_ID	int	Identifikator der Prozedur	Nein

Tabelle FILE_FO_ZUORDNUNG Wie bereits erwähnt besitzt jedes Sprachsignal ein bestimmtes Format. Diese Tabelle stellt die Zuordnung zwischen einem Sprachsignal und dessen Format dar.

Feldname	Typ	Bedeutung	Leer
ID	int	Identifikator der Signaldatei	Nein
FL_FO_ZU_ID	int	Identifikator des Formats	Nein

Tabelle FILE_HT_ZUORDNUNG Die Beziehung zwischen eines Sprachsignals und seines Haupttyps wird in dieser Tabelle festgelegt.

Feldname	Typ	Bedeutung	Leer
ID	int	Identifikator der Signaldatei	Nein
FL_HT_ZU_ID	int	Identifikator des Haupttyps	Nein

Tabelle FILE_UT_ZUORDNUNG Die Beziehung zwischen eines Sprachsignals und seines möglichen Untertyps wird in dieser Tabelle festgelegt.

Feldname	Typ	Bedeutung	Leer
ID	int	Identifikator der Signaldatei	Nein
FI_UT_ZU_ID	int	Identifikator des Untertyps	Nein

Tabelle FILE_QU_ZUORDNUNG Jede Sprachsignaldatei stammt aus einer bestimmten Quelle. Die Information über die Quelle wird bereits in der Tabelle FILE gespeichert. Da diese Tabelle die Beziehung zwischen einer Signaldatei und der entsprechenden Quelle darstellt, ist sie gewissermaßen redundant. Es ist in diesem Fall auch möglich, daß die Tabelle FILE_QU_ZUORDNUNG im Laufe der Programmentwicklung nicht mehr benötigt wird.

Feldname	Typ	Bedeutung	Leer
ID	int	Identifikator der Signaldatei	Nein
FI_QU_ZU_ID	int	Identifikator der Quelle	Nein

Tabelle FILE_ZUG_ZUORDNUNG Diese Tabelle drückt durch die Verknüpfung zwischen einer Signaldatei und einem Zugriffsrecht die Zugriffsberechtigung hinsichtlich dieser Signaldatei aus. Es ist gegenwärtig allerdings noch nicht endgültig festgelegt, wie sich die Zugriffsberechtigung auf den Programmablauf insgesamt auswirkt. Aus diesem Grund hat diese Tabelle nur eine vorläufige Struktur vorzuweisen.

Feldname	Typ	Bedeutung	Leer
ID	int	Identifikator der Signaldatei	Nein
FI_ZUG_ZU_ID	int	Identifikator des Zugriffsrechts	Nein

1.7 Beziehungen zwischen den Tabellen

Folgende Anforderungen werden an das Datenbankmodell gerichtet:

- es soll in Bezug auf die Tabellenstruktur beliebig erweiterbar sein, also möglichst offen sein.
- es soll einen möglichst schnellen, komfortablen Zugriff auf die Daten gewährleisten.
- es soll die Daten im konsistenten Zustand aufrechterhalten, also entsprechende Mechanismen zur Verfügung stellen, die die referentielle Integrität kontrollieren.
- es soll handhabbare Operationen ermöglichen (Daten eintragen, Daten ändern, Daten löschen ...).

Es ist möglich, alle Informationen über Sprachsignaldateien mitsamt ihren Formaten, Haupttypen, Untertypen, Quellen usw. in eine einzige Tabelle zusammenzupacken. Dabei sind Haupttypen, Untertypen und Formate beispielhafte Exemplare von erweiterten Eigenschaften der Sprachsignaldateien zu interpretieren, die in der Datenbank erfaßt werden sollen. In diesem Fall braucht man nur auf diese Tabelle zuzugreifen, der Zugriff beschränkt sich also nur auf diese betroffene Tabelle und man braucht nicht um die Beziehungen zwischen den Tabellen zu kümmern. Diese Variante bringt aber einen schwerwiegenden Nachteil mit sich. Das Datenbankmodell läßt sich nicht so leicht erweitern, der Grad der Offenheit wird also ohne erforderliche Maßnahmen eingeschränkt. Überdies erfordert die Eintragung eines Datensatzes auf einmal viele Informationen, die vor der Einfügung gesammelt werden müssen. So gesehen ist beispielsweise die Eintragungsoperation (INSERT) nicht so sehr benutzerfreundlich.

Von den Anforderungen und dem oben genannten Nachteil ausgehend werden nur die unmittelbaren relevanten Informationen einer Sprachsignaldatei zentralisiert gehalten, weitere Eigenschaften werden dezidiert über mehrere sogenannte **Satellitentabellen** verteilt. Zwischen der Zentraltabelle und den Satellitentabellen sind unbedingt Verknüpfungen zu realisieren und aufrechtzuerhalten. Explizite Beziehungen zwischen Tabellen werden bereits in den oben vorgestellten Beziehungstabellen ausgedrückt. Auf diese Weise wird die oben erwähnte Eintragungsproblematik elegant gelöst. Es existieren zwei Kategorien von Routinen zum Eintragen von Sprachsignaldaten in die Datenbank. Bei der ersten Kategorie werden die unmittelbaren relevanten Informationen wie z.B. Dateiname, Dateiextension, Dateipfad ... in die Tabelle FILE eingetragen. Bei der zweiten Kategorie werden "erweiterte Eigenschaften" wie z.B. Haupttyp, Untertyp, Formate einer Sprachsignaldatei in den entsprechenden Tabellen registriert (FILE_HT_ZUORDNUNG, FILE_UT_ZUORDNUNG, FILE_FO_ZUORDNUNG).

1.8 Bemerkungen zum Datenbankmodell und Datenbanksystem

Nach der ausführlichen Beschreibung der Tabellenstruktur und der Handhabung von mitgelieferten administrativen Tools ist es angebracht, auf einige charakteristische Merkmale des Datenbankmodells hinzuweisen.

Die die Zentralrolle spielende Tabelle ist die Tabelle FILE, die relevante Informationen über Sprachsignaldateien enthält. Da das Datenbankmodell mit der Tabelle FILE verbunden sind. Auf diese Weise kann auf die strukturierten Informationen schnell zugegriffen werden. Auch die Integration von neuen Objekten und die Generierung und Verwaltung von neuen Transformationsprozeduren werden dadurch unterstützt, weil neue Objekte in die "Satellitentabellen" eingefügt zu werden brauchen. Somit kommt das verwendete Datenbankmodell am bestens zur Geltung. In Bezug auf das Datenbankmodell sollen folgende Sachverhalte auf jeden Fall beachtet werden:

- Es liegt in der Verantwortung des Benutzers, die Daten konsistent zu halten bei datenmanipulierenden Operationen (Einfügen, Löschen), weil das verwendete mSQL-System Release 1.06 keine entsprechenden Mechanismen zur Verfügung stellt, wie z.B. Propagation beim Löschen oder eingebaute Regel zum Entdecken von Verletzungen der referentiellen Integrität. Im C-Programm werden ersatzweise solche Mechanismen bereits eingebaut, beispielsweise werden Datensätze in der Tabelle FILE_PRO_ZUORDNUNG beim Löschen einer Datei in der Tabelle FILE mitgelöscht, sofern die gelöschte Datei in den mitgelöschten Datensätzen der Tabelle FILE_PRO_ZUORDNUNG enthalten ist.
- Es wird versucht, die Zugriffsrechte auf eine Sprachsignaldatei softwaremäßig (nicht mit Hilfe vom UNIX-Zugriffsmechanismus) zu regeln. Aus diesem Grund wird die Tabelle FILE_ZUG_ZUORDNUNG erstellt. Auch werden entsprechende Routinen zur Verwaltung der Zugriffsrechte und deren Zuordnungsmöglichkeit implementiert, die hauptsächlich mit dieser Tabelle operieren. Der Entwicklungsstand ist jedoch nicht ausgereift, so dass die Tabelle FILE_ZUG_ZUORDNUNG momentan nur eine statische Rolle spielt.
- Wie bereits erwähnt bietet die verwendete Version des Datenbanksystems mSQL keine geeignete Datentypen zum direkten Speichern von Daten von großem Umfang oder im Binär-Format (siehe 1.4). Beispielsweise kann eine SD-Sprachsignaldatei nicht direkt in der Datenbank abgespeichert werden. Aus diesem Grund werden nur Namen der Sprachsignaldateien in der Datenbank in der Tabelle FILE registriert. Unter einem bestimmten Gesichtspunkt kann dies als Einschränkung betrachtet werden. Allerdings ist man bei diesem Vorgehen flexibler bei der Verwaltung und Pflege der Daten. Obendrein ist die Datenbank nicht so sehr aufgebläht und nimmt weniger Plattenkapazität in Anspruch.

- Die Haupttabelle FILE weist eine Struktur auf, die sich erheblich an der Bezeichnung und der Zusammensetzung der vorliegenden Sprachsignaldateien orientiert (wie z.B. dlf0224.1656.sd), weil es sinnvoll erscheint, auf die einzelnen Bestandteile der Bezeichnung zugreifen zu können. Überdies ermöglicht diese Strukturierung verschiedene Aspekte der gezielten Anfragen wie z.B. Anfragen nach Namen, Datum, Extension u.s.w.
- Es ist unter Umständen erforderlich, eine leere Korpora-Datenbank zu kreieren. Dazu ist die ausführbare Datei "korpodb" zu verwenden, deren Aufruf nur erfolgreich ist, wenn die Vorlagedatei "signaltemplate.dmp" im selben Verzeichnis zu finden ist. An dieser Stelle ist es anzumerken, daß die Vorlagedatei "signaltemplate.dmp" eine Dump-Datei einer Datenbank ist, die als Vorlage dient. Sie enthält die Struktur der in der Datenbank definierten Tabellen. Unter Umständen ist es allerdings gewünscht, den aktuellen Datenbestand der Datenbank in einer Dump-Datei festzuhalten oder Daten von einer bestehenden Dump-Datei in eine nicht leere Datenbank zu importieren. Dies kann ebenfalls mit Hilfe des Tools "msqldump" und der Importroutine im C-Programm bewerkstelligt werden.
- Die Datenbanken werden automatisch von mSQL erzeugt und unter dem Verzeichnis "msqldb" abgelegt. Jede Datenbank besteht aus einer Reihe von Dateien verschiedener Extensionen, die wiederum im Binär-Format vorliegen. Die Verzeichnisstruktur der Datenbank ist also festgelegt.
- Mit dem Datenbanktool "msqldump" können Dump-Dateien erzeugt werden, die die aktuellen in der Datenbank enthaltenen Daten repräsentieren. Statt der Bezeichnung Dump-Datei kann man ebenso die Bezeichnung Backup-Datei verwenden, denn man kann mit der Dump-Datei eine komplette neue Datenbank erzeugen oder Daten nachträglich in eine vorhandene Datenbank hinzufügen (Datenbankpopulation). Es ist sogar zu empfehlen, regelmäßig eine Dump-Datei von der verwendeten Datenbank zu erstellen, die im Notfall als Recovery-Datei wieder eingesetzt wird.

Kapitel 2

Programmaufbau

Aufgrund einer der Anforderungen an die Implementierung, daß sie möglichst offen für Erweiterungen sein soll, wird der Programmaufbau in Anlehnung daran programtechnisch ebenfalls flexibel gestaltet. Die Implementierung besteht im wesentlichen aus zwei Hauptbestandteilen:

- dem C-Verwaltungsprogramm
- und der Perl-Auswertungsschnittstelle

Das C-Programm hat einen modularen Aufbau, stellt Funktionen zur Verfügung, die unmittelbar mit den Daten aus der Datenbank operieren, und dient somit in erster Linie der Verwaltung der Daten in der Datenbank.

Die Perl-Schnittstelle nimmt Anfrageergebnisse in Form von Dateilisten vom C-Verwaltungsprogramm entgegen und führt damit die Auswertung der Ergebnisse durch.

2.1 Programmmodule in C

Das C-Programm besteht aus insgesamt 8 C-Modulen, die wie folgt genannt sind:

- main.c
- hilfe1.c
- hilfe2.c
- hilfe3.c
- hilfe4.c
- hilfe5.c
- hilfe6.c
- hilfe7.c

main.c

Dieses Modul ist das Hauptmodul, enthält u.a. die obligatorische Einstiegsfunktion `main()` eines C-Programms, Abfragefunktionen (`SELECT`) und Funktionen zur Abarbeitung der möglich vorhandenen Kommandozeileoptionen (`SELECT`). In diesem Modul wird auch die Verbindung zum Datenbankserver hergestellt, gesetzt den Fall, daß der Datenbankserver bereits gestartet wurde. An dieser Stelle ist es geeignet zu empfehlen, den Datenbankserver beim Hochfahren des Host gleich zu starten, wo `mSQL` installiert wird.

Nach erfolgreicher Verbindung zum Datenbankserver erscheinen auf dem Bildschirm im Regelfall (partiellen Automatisierungsmodus) verschiedene Menüpunkte, die zu einem bestimmten Zweck ausgewählt werden können. Auf Seite 121 werden die wichtigsten Routinen von `main.c` aufgelistet.

hilfe1.c

Dieses Modul enthält u.a. einige Hilfsroutinen zum Ermitteln eines eindeutigen neuen Identifikators, zum Eingeben von Informationen wie Dateinamen, Datum, Uhrzeit, Extension, Quelle und Einfügeoperationen zum Eintragen der Daten in die entsprechenden Tabellen (`INSERT`). Die in diesem Modul enthaltenen Routinen sind auf Seite 122 aufgelistet.

hilfe2.c

Dieses Modul enthält u.a. Funktionen, die zum Löschen (`DELETE`) der Daten in den Tabellen herangezogen werden. Wichtige Exemplare von Routinen dieses Moduls sind auf Seite 123 zu finden.

hilfe3.c

In diesem Modul sind Funktionen zu finden, die Daten von einer Inputdatei einliest und in die Tabelle `FILE` einträgt oder Daten von der Tabelle `FILE` löscht. Mit Hilfe dieser Funktionen kann man auf bequeme Weise mehrere Datensätze auf einmal (also in Batchform) eintragen oder löschen. Die Funktionen sind auf Seite 124 zu finden.

hilfe4.c

Dieses Modul enthält u.a. Funktionen zum Ändern (`UPDATE`) von in der Datenbank vorliegenden Daten, welche auf Seite 124 aufgelistet sind.

hilfe5.c

In diesem Modul finden sich weitere Funktionen, die beispielsweise Informationen über Anzahl der Datensätze innerhalb einer Tabelle zurückliefern. Es handelt sich hierbei um zusätzliche Abfragemöglichkeiten, die den Benutzern zur Verfügung stehen (siehe Seite 125).

hilfe6.c

Dieses Modul enthält Funktionen (siehe Seite 126) zur Umwandlung von Daten eines Formats in ein anderes Format, wie z.B. von `"Sd"` nach `"Sph"`, `"Sd"` nach `"Phones"`. Die Umwandlung kann wahlweise entweder interaktiv (durch Angabe einer umzuwandelnden Datei) als auch in Batch-Form (durch Angabe einer Inputdatei) erfolgen.

hilfe7.c

In diesem Modul sind Hilfsroutinen untergebracht, die bei der Abwicklung der Kommandozeileoption-Überprüfung sowie -Erkennung eingesetzt werden (siehe Seite 126).

2.2 Programmmodul in Perl

Wie bereits erwähnt ist eine der Anforderungen in der Aufgabenstellung ist die flexible Auswertung der Abfrageergebnisse, welche vom C-Programm zurückgeliefert werden. Bei dieser Thematik wird ein anderer Weg eingeschlagen.

Zur Analyse von Sprachsignalen werden die Abfrageergebnisse in eine ASCII-Datei gespeichert, die wiederum als Inputdatei für ein Perl-Programm namens **analyse.pl** fungiert. Dieses Perl-Programm durchsucht die Inputdatei nach bestimmten Wörtern oder Phones und bringt seinerseits eine Output-Datei hervor. Dem Perl-Programm können Kommandozeileoptionen übergeben werden. U.a. kann ein Programmaufruf als Kommandozeile angegeben werden, der in der Outputdatei mit gespeichert wird. So kann man auf eine sehr flexible Weise ein anderes Programm ausgehend von der mit Kommandozeilen formatierten Outputdatei starten. Genau dies ist der Clou dieser Auswertungsmethode.

Wichtige Eigenschaften des Programms `analyse.pl` sind zusammengefaßt wie folgt :

- logische Ausdrücke von Elementen (Lauten, Silben)
- frei definierbare Features
- logische Ausdrücke von Features
- relative Segmentadressierung
- flexible Formatierung der Ausgabedatei

Detaillierte Beschreibung der Perl-Schnittstelle wird im Abschnitt 4.1 behandelt.

Kapitel 3

Ablauf des C-Programms

Um die Anwendung beherrschen zu können, soll dem Benutzer zumindest erklärt werden, wie das Verwaltungsprogramm gestartet wird, welche funktionale Möglichkeiten es insgesamt gibt und wie der Programmverlauf aussieht. In diesem Abschnitt wird ansatzweise der Versuch gemacht, dem Benutzer die einzelnen Programmabläufe näher zu bringen.

3.1 Programmstart

Im Prinzip kann das C-Verwaltungsprogramm in zwei Modi gestartet werden: partielle Automatisierung und vollständige Automatisierung. Der Begriff "Automatisierung" suggeriert unter diesem Aspekt einen eigenständigen Programmablauf ohne Einwirkung des Benutzers. In diesem Sinne muß der Benutzer bei der partiellen Automatisierung gegebenenfalls Inputwerte interaktiv eingeben, während er bei der vollständigen Automatisierung von vorne herein das Verwaltungsprogramm mit entsprechenden Inputwerten in Form von mit übergebenen Argumenten "füttert" und das Ganze in Form von Kommandobefehlen absetzt.

Der erstgenannte Modus birgt in sich den Vorteil, daß der Benutzer die vollständige Kontrolle bei der Datenbankadministration hat und das Ergebnis der zuvor ausgeführten Datenbankoperation überprüfen und ansehen kann. Im Gegensatz dazu ist der zweite Modus von der Ablauflogik her für den Fall geeignet, wenn von anderen Anwendungen heraus eine Routine im Verwaltungsprogramm gestartet wird.

3.1.1 Partielle Automatisierung

Das C-Programm wird durch die Angabe des Kommandos "korpo" gestartet und ist menügesteuert. Die Menüpunkte sind in mehreren Ebenen untergliedert und baumartig aufgebaut. Jede Ebene ist wiederum jeweils mit einem Buchstaben, beginnend bei A, assoziiert. Ebene A ist das Hauptmenü und kommt zuerst in Erscheinung, sofern das Programm ohne Kommandozeileoption gestartet wird. Jeder Menüpunkt ist ebenfalls jeweils mit einer Nummer gekennzeichnet, die die Position dieses Menüpunktes innerhalb seiner Ebene darstellt. Die Menüpunkte und die entsprechenden Funktionen können also auf zwei Arten aktiviert werden:

- interaktiv durch Eingabe einer entsprechenden Kennzeichnungsnummer
- per Kommandoswitch. Wobei besteht ein Kommandoswitch aus dem obengennanten Buchstaben, der die Ebene des Menüpunkts darstellt, und der den Menüpunkt kennzeichnenden Nummer. Diese zwei Größen werden durch das Gleichheitszeichen miteinander verknüpft. Durchs

Aneinanderreihen von entsprechenden Kommandozeileoptionen, angefangen von der obersten Ebene der Hierarchie (Ebene A) bis zum Zielebene, kann man sofort zur Zielebene springen. Dabei werden die jeweils eine Ebene repräsentierende Kommandozeileoptionen durch ein Leerzeichen getrennt.

Bei unvollständiger Angabe der Kommandozeileoption erreicht man einen Zwischenknoten innerhalb des vordefinierten Menüsystems. In diesem Fall wird die entsprechende Menüebene angezeigt und es wird automatisch zum interaktiven Modus gewechselt. Der Benutzer muß anschließend eine Nummer eingeben, um zu seinem Ziel zu gelangen.

Beispiel:

Ebene A	1:	Abfragen
	2:	Datei umwandeln
	3:	Datenbankroutinen: Eintragen
	4:	Datenbankroutinen: Löschen
	5:	Datenbankroutinen: Änderungen
	0:	Beenden

Um Menüpunkt "Datenbankroutinen: Eintragen" zu aktivieren, kann man wie folgt vorgehen:

- interaktiv: *korpo* < RETURN > 3 < RETURN >
- mit Kommandozeileoption: *korpo A = 3* < RETURN >

Ebene A1_B	1:	Datei-Abfragen
	2:	Prozedur-Abfragen
	3:	Format-Abfragen
	4:	Haupttyp-Abfragen
	5:	Untertyp-Abfragen
	6:	Quelle-Abfragen
	7:	Zugriffsrecht-Abfragen

Um Ebene A1_B wie oben dargestellt zu aktivieren, kann man wie folgt vorgehen:

- interaktiv: *korpo* < RETURN > 1 < RETURN > 1 < RETURN >
- mit Kommandozeileoption: *korpo A = 1 B = 1* < RETURN >

Es ist hierbei zu beachten, daß drei Kommandozeileoptionen "-i", "-s" und "-r" besonderen Augenmerk genießen.

- Die Kommandozeileoption [-i] leitet eine eingebaute Hilfe ein. Wenn die Angabe " korpo -i " lautet, wird eine Liste angezeigt, in der alle Kommandozeileoptionen zur Auswahl der Menüpunkte aufgelistet sind. Mit Ja kann die Liste der Hilfe weiter geblättert werden, mit Nein wird das Programm beendet.
- Die Kommandozeileoption [-s] gibt den Namen der Datenbank an, mit der das Programm zusammenarbeiten soll. Wenn diese Option nicht angegeben ist, wird automatisch versucht, zur Datenbank "korpora" eine Verbindung aufzubauen. Der Aufruf lautet in diesem Fall beispielsweise *korpo -s "test"*. Vorweggenommen ist die Datenbank "korpora" eine neue Datenbank, mit der während der Testphase gearbeitet wird. Die Beschreibung der Erstellung dieser Datenbank wird später ausführlich dargestellt. Mit dieser Kommandozeileoption kann man abwechselnd mit unterschiedlichen Datenbank arbeiten. Dies ist beispielsweise besonders wertvoll beim Testen einer neuen Datenbank.

- Die Kommandozeileoption [-r] ermöglicht die Anbindung zu einem Remote- Datenbankserver, der durch einen Hostnamen oder eine IP-Adresse repräsentiert wird. Diese Kommandozeileoption ist hilfreich, sofern man von einem Rechner auf die Datenbank zugreift, die woanders (auf einem anderen Rechner) installiert ist. Der Aufruf lautet wie z.B. *korpo -r "ims"*.
- Wenn keine Angaben über Hostnamen (mittels [-r]) und/oder über die anzubindende Datenbank gemacht werden, dann wird automatisch versucht, an die Datenbank namens korpora auf dem lokalen Rechner anzubinden.

Die zwei Kommandozeileoptionen [-s] und [-r] können in beliebiger Reihenfolge zusammen angegeben werden. Dies wird in diesem Fall interpretiert als der Versuch zur Anbindung an die durch [-s] spezifizierte Datenbank auf dem durch [-r] spezifizierten Hostrechner. Es wird ebenfalls überprüft, ob die Kommandozeileoptionen ordnungsgemäß angegeben werden. Überdies können die zwei Kommandozeileoptionen [-r] und [-s] zusammen mit anderen Kommandoswitchs zum Einstiegen in ein Menüsystem angegeben werden.

Beispiel:

- *korpo -r "sperber" -s "test" A=1 B=1*
- *korpo -s "test" A=1 B=1 C= 1*
- *korpo -r "sperber" A=1*
- *korpo A=1 B= 1*

Je nach Aufruf wird stets zunächst versucht, eine Verbindung zum Datenbankserver herzustellen und die angegebene Datenbank (per Kommandozeileoption oder per Voreinstellung) anzubinden. Dies geschieht innerhalb der im C-Verwaltungsprogramm implementierten Funktion `db_connect()`, die bekanntlich zwei als Zeichenketten zu übergebende Parameter hat: den Datenbanknamen und den Namen des Remote- Datenbankservers. Diese Funktion bedient sich ihrerseits der eingebauten API-Funktion `mysqlConnect` und liefert wie bereits erwähnt einen Datenbankhandle namens "dbsock" zurück, der während der gesamten Sitzung immer als Parameter für weitere Zugriffsfunktionen benutzt wird. Die Funktion `db_connect` wird wie folgt definiert:

```
int db_connect(char* dbname, char *remote_dbsrvr)
```

Wenn `db_connect()` nicht fehlschlägt, steht die Verbindung zum Datenbankserver und die gewünschte Datenbank wird fehlerfrei angebunden. Danach erscheint eine Reihe von Hauptmenüpunkte auf dem Bildschirm, die als Einstieg in das gesamte Menüsystem dienen. Im folgenden Abschnitt wird eine Übersicht des Menüsystems zur Illustration angezeigt.

3.1.2 Vollständige Automatisierung

In diesem Modus kehrt die Programmkontrolle sofort zurück zur aufrufenden Anwendung, sofern die innerhalb des Verwaltungsprogramms aufgerufenen Funktion beendet ist. Auf diese Weise können vom Verwaltungsprogramm zur Verfügung gestellte Funktionalitäten von anderen Anwendungen heraus eingesetzt und verfügbar gemacht werden. Dabei sind die vom vorhergehenden Abschnitt beschriebenen Kommandozeileoptionen unbedingt erforderlich. Außerdem müssen dieselbe Angaben über erforderliche obligatorische Inputwerte gemacht werden, als wenn man solche Werte interaktiv eingeben würde. Das abzusetzende Kommando wird in der Regel durch die Option [-e] abgeschlossen, damit das Verwaltungsprogramm beendet (terminiert) wird. Ohne die Option [-e] am Schluß findet man sich im interaktiven Modus des Verwaltungsprogramms wieder. Dabei ist es unbedingt zu beachten, daß die Option [-e] immer am Ende der ganzen Kommandokette angegeben wird. Folgendes Beispiel verdeutlicht die verschiedenen Startarten, indem die unterschiedlichen Angaben zum Vergleich gegenübergestellt werden:

- Partielle Automatisierung

- ohne Kommandozeileoption

```
korpo -s korpora, 3 <RETURN>, 5 <RETURN>, <Filename>
```

- Mit Kommandozeileoption

```
korpo -s korpora A=3 B=5, <Filename>, eingeben nach Aufforderung
```

- Vollständige Automatisierung

```
korpo -s korpora A=3 B=5 <Filename> -e
```

3.2 Wichtige Vorbemerkungen zum Verwaltungsprogramm

Bevor eine ausführliche Beschreibung der einzelnen Menüpunkte dargeboten und deren Funktionsweise erläutert wird, werden in diesem Abschnitt vorweggenommen einige Vorbemerkungen beschrieben, die als Anhaltspunkte dienen und das Verstehen der menügesteuerten Programmführung im allgemeinen fördern sollen.

3.2.1 Allgemeine Aspekte

In Bezug auf die gewünschte flexible Gestaltung der Anfragen ist die Implementierung der Anfragen innerhalb eines C-Programms keine optimale Lösung, obwohl es versucht wird, von vorne herein all die möglichen Fälle der spezifischen Anfragen vorauszuplanen und die auftretenden Parameter zu untersuchen und zu berücksichtigen. Das C-Programm weist aus diesem Grund einen beträchtlichen Umfang von diversen Hilfsfunktionen auf zur Datenbankadministration und zur Datenverwaltung. Was ganz wichtig ist, ist die Tatsache, daß die in der Aufgabenstellung manifestierte Funktionalität gewährleistet wird.

Falls neue Anfragen in Zukunft erforderlich sind, muß man solche Anfragen in Form von C-Routinen formulieren und das C-Programm neu kompilieren, damit die neuen Funktionen gültig werden. Der Kompilervorgang beschränkt sich ledig auf den Aufruf der entsprechenden Makefile. Trotzdem muß dieser Vorhang jedesmal wiederholt werden, sofern der Funktionsumfang des C-Programms erweitert wird. Dies aber fällt nicht so sehr ins Gewicht, weil das C-Programm so angelegt ist, daß möglichst notwendige Abfragen abgedeckt sind. Überdies erfolgt die Suche nach dem Suchmuster in den Anfragen immer case-sensitiv.

Anfragen, die auf Daten der Haupttabelle FILE bezogen sind, sind immer mit einer Ergebnisdatei verbunden, die für mögliche anschließende Analyse durch die Perl-Schnittstelle als Inputdatei hergehalten wird. Diese Inputdatei wird im Verzeichnis abgespeichert, von wo das C-Programm gestartet wird. Falls der Benutzer für dieses Startverzeichnis kein Schreibrecht besitzt, erscheint eine entsprechende Fehlermeldung und die Anfrage wird dementsprechend abgebrochen.

Wie bereits mehrfach erwähnt erfolgt das Zusammenspiel zwischen den zwei Hauptprogrammteilen, dem Verwaltungsprogramm in C und dem Analyseprogramm in Perl, im Normalfall über eine Outputdatei, die Anfrageergebnisse enthält und vom Verwaltungsprogramm erstellt wird. Aus diesem Grund wird bei allen Menüpunkten, in denen Anfrage an den Datenbankserver abgesetzt werden können, eine Outputdatei erzeugt, die im partiellen Automatisierungsmodus vom Benutzer benannt werden kann, im vollständigen Automatisierungsmodus aber immer "erg.dat" heißt.

3.2.2 Löschen von Daten aus den Tabellen

Beim Löschen von Sprachsignal-Datensätzen aus der Tabelle FILE in Batchform sowie beim Einlesen von Daten in dieselbe Tabelle in Batchform wird eine Inputdatei benötigt, die in der Regel unter demselben Verzeichnis zu finden ist, wo sich das C-Programm ebenfalls befindet. Ansonsten muß die Inputdatei mit kompletter Pfadangabe angegeben werden.

Im Gegensatz zum Löschen von Daten aus der Tabelle FILE ist es nicht möglich, eine Tabelle vom C-Programm aus mittels des üblichen Platzhalters "*" komplett zu löschen. Nur Dateneinträge, die durch eine entsprechende, obligatorische Angabe spezifiziert sind, werden gelöscht. Auf diese Weise kann das C-Programm die Propagation des Löschvorgangs an andere Tabellen weiter leiten und kontrollieren, die eine Beziehung mit der ursprünglichen Tabellen haben, weil das mSQL-Datenbanksystem wie bereits erwähnt das kaskadierte Löschen nicht unterstützt. Unter Umständen kann dies als eine Einschränkung seitens des C-Programms angesehen werden. Die Implementierung des kaskadierten Löschen für den Platzhalter "*" ist kompliziert und auch nicht sinnvoll für andere Tabellen. Soll eine Tabelle dennoch komplett gelöscht werden, kann man sinnvollerweise auf den mitgelieferten Tool "msql" zurückgreifen und entsprechende SQL-Statement ausführen lassen.

Beispiel

```
Tabelle Haupttyp wird komplett gelöscht
msql korpora
delete * from haupttyp\g
```

Falls die Tabelle FILE_HT_ZUORDNUNG Daten enthält, die einen gelöschten Haupttyp haben, müssen sie auch gelöscht werden (kaskadiertes Löschen), damit die Daten in der Datenbank konsistent bleiben.

3.2.3 Formatumwandlungen

Bei Formatumwandlungen werden vordefinierte Transformationsprozeduren direkt vom C-Programm aus aufgerufen. Somit hängt die Formatumwandlung zum größten Teil von diesen vordefinierten Transformationsprozeduren ab. In erster Linie gilt es, die Umgebungsvariable "**ExPho**" richtig einzustellen. Diese Systemvariable gibt den Zielpfad an, also den Ort, wo die umgewandelte Sprachsignaldatei nach der Umwandlung abgelegt werden soll. Die Definition dieser Variablen kann in der benutzerabhängigen Startdatei ".cshrc" (unter C-Shell) vorgenommen werden. Überdies gestaltet sich der Aufruf der Transformationsprozeduren hinsichtlich der Argumente in Abhängigkeit von ihrer Implementierung. Beispielsweise erfordert die Prozedur "e2sphere" den Namen einer Outputdatei, während Alignphones und Alignwords darauf verzichten. Die Formatumwandlung kann aber nicht im Hintergrund gestartet werden, sofern man sich des C-Programms bedient.

Eine Bemerkung, die unbedingt an dieser Stelle noch mal in Erinnerung gerufen werden muß, ist, daß das Ergebnis der Formatumwandlung (der Name der umgewandelten Datei samt einigen Informationen) jeweils in den temporären Tabellen (FILE_SPH, FILE_PHONES ...) gespeichert wird. Anhand dieser Tabellen kann ermittelt werden, ob sich eine Datei bereits der Formatumwandlung unterzogen hat. Somit dienen diese Tabellen der zwecksgebundenen Kontrolle. Im Laufe der Zeit können diese Tabellen allerdings hinsichtlich ihrer Größe unkontrolliert wachsen. Es ist aus diesem Grund ratsam, diese Tabellen von Zeit zu Zeit zu löschen. Dafür wird die Routine delete_tabelle_allgemein() aufgerufen, der der Name der zu löschenden Tabelle übergeben wird. Die Routine löscht eine Tabelle komplett. Aus diesem Grund soll der Umgang mit ihr immer von Vorsicht begleitet sein.

Bei der Umwandlung nach Sph-Format wird die Transformationsprozedur `e2sphere` gestartet. Als Ergebnis liegt die Sph-Datei im Verzeichnis `/usr/local/tmp/ExPhoDB` vor.

Bei der Umwandlung nach Phones-Format werden in der Regel die entsprechenden Sph- und Mfc-Dateien zuerst erzeugt und im Verzeichnis `/tmp` abgelegt. Die Erstellung der Mfc-File ist dabei sehr zeitintensiv. Die Phones-Datei wird u.a. im Verzeichnis `/usr/local/tmp/ExPhoDB` abgelegt (es entstehen dabei auch die Phonemic- und PhoneswithQ-Dateien). In diesem Fall wird die Transformationsprozedur `Alignphones` eingesetzt.

Bei der Umwandlung nach Words-Format wird die Transformationsprozedur `Alignwords` gestartet. Der Vorgang ähnelt sich dem bei der Formatumwandlung nach Phones-Format.

Bei der extensionsbezogenen Formatumwandlung kann derzeit nur das Format Sd berücksichtigt werden. D.h., bei Angabe anderer Formate wird die zuständige Routine sofort beendet. Dies kann eine sehr restriktive Maßnahme sein. Der Grund dafür besteht darin, das im Moment nur die Transformationsroutinen `Alignphones`, `Alignwords`, `e2sphere` zur Verwendung kommen, die ja nur mit Sd-Sprachsignaldateien umgehen können. Im Zusammenhang mit den Formatumwandlungen soll die Einstellung der Umgebungsvariablen "ExPho" allerdings vorerst vorgenommen werden. Einzelheiten dafür finden sich im Abschnitt 7.3.

3.2.4 Datenimport in Batchform

Mit der Kommandozeileoption `A=3B=1C=2` können Daten aus einer zuvor präparierten Datei in die Tabelle FILE übertragen werden, ohne daß der Benutzer dabei interaktiv in das Geschehen eingreift. Jeder Dateneintrag in der Tabelle FILE wird durch eine ID-Nummer eindeutig identifiziert. Die Übertragungsroutine prüft anhand des Namens der zu importierenden Sprachsignaldatei zuerst, ob diese Datei bereits in der Datenbank vorhanden ist. Falls es nicht der Fall ist, wird eine neue ID-Nummer ermittelt und dieser Datei zugewiesen. Wenn der Datenbestand der Datenbank umfangreicher wird, dauert dieser Vorgang dementsprechend etwas länger.

3.3 Menüsystem

Dieser Abschnitt gibt einen programmtechnisch detaillierten Überblick über die mehrfach erwähnte menügesteuerte Programmführung sowie die dabei aktivierten Routinen.

1:	Abfragen	Untermenü
2:	Datei umwandeln	<code>ab_file_umwandeln(dbsock,argc,argv)</code>
3:	Datenbankroutinen: Eintragen	Untermenü
4:	Datenbankroutinen: Löschen	Untermenü
5:	Datenbankroutinen: Änderungen	Untermenü
0:	Beenden	

Je nach gewähltem Menüpunkt gelangt man zu weiteren Untermenüpunkten auf tieferen Ebenen. Am Schluß wird eine entsprechende Funktion ausgeführt, wenn der betroffene Menüpunkt nicht mehr verzweigt. Beispielsweise wird die Funktion `ab_file_umwandeln(dbsock,argc,argv)` unmittelbar ausgeführt, wenn der Menüpunkt "Datei umwandeln" ausgewählt wird.

3.3.1 Hauptmenüpunkt Abfragen

Dieser Menüpunkt führt zu folgenden Untermenüpunkten:

1:	Datei-Abfragen	Untermenü
2:	Prozedur-Abfragen	Untermenü
3:	Format-Abfragen	Untermenü
4:	Haupttyp-Abfragen	Untermenü
5:	Untertyp-Abfragen	Untermenü
6:	Quelle-Abfragen	Untermenü
7:	Zugriffsrecht-Abfragen	Untermenü
0:	Zurück	

Untermenüpunkt Datei-Abfragen

Dieser Untermenüpunkt gliedert sich in folgende Menüpunkte auf:

1:	Alle Dateien	ab_file_alle(dbsock,int)
2:	Datei-Prozedur-Zuordnung	ab_file_zugeordnet(dbsock,int)
3:	Datei-Haupttyp-Zuordnung	ab_file_haupttyp_zugeordnet(dbsock,int)
4:	Datei-Untertyp-Zuordnung	ab_file_untertyp_zugeordnet(dbsock,int)
5:	Dateiauswahl nach Namen	ab_file_name_auswahl(dbsock,argc,argv)
6:	Dateiauswahl nach Stamm	ab_file_pre_auswahl(dbsock,argc,argv)
7:	Dateiauswahl nach Datum	ab_file_datum_auswahl(dbsock,argc,argv)
8:	Dateiauswahl nach Typ (Endung)	ab_file_endung_auswahl(dbsock,argc,argv)
9:	Dateiauswahl nach Quelle	ab_file_quelle_auswahl(dbsock,argc,argv)
10:	Dateiauswahl nach Pfad	ab_file_pfad_auswahl(dbsock,argc,argv)
11:	Dateianzahl	Untermenü
12:	Alle umgewandelte Dateien	ab_file_output(dbsock,argc,argv)
0:	Zurück	

Wobei:

- `ab_file_alle(dbsock,int)` : listet alle Signaldateien in der Datenbank auf.
- `ab_file_zugeordnet(dbsock,argc,argv)`: listet alle Signaldateien auf, die einer Prozedur zugeordnet sind. Eine Signaldatei ist einer Prozedur zugeordnet, wenn der Identifikator der Signaldatei in der Tabelle `FILE_PRO_ZUORDNUNG` zu finden ist.
- `ab_haupttyp_zugeordnet(dbsock,int)`: listet alle Signaldateien auf, die einem Haupttyp zugeordnet sind. Eine Signaldatei ist einem Haupttyp zugeordnet, wenn der Identifikator der Signaldatei in der Tabelle `FILE_HT_ZUORDNUNG` zu finden ist.
- `ab_untertyp_zugeordnet(dbsock,int)`: listet alle Signaldateien auf, die einem Untertyp zugeordnet sind. Eine Signaldatei ist einem Untertyp zugeordnet, wenn der Identifikator der Signaldatei in der Tabelle `FILE_UT_ZUORDNUNG` zu finden ist.
- `ab_file_name_auswahl(dbsock,argc,argv)`: listet Signaldateien auf mit dem gesuchten Namen.
- `ab_file_pre_auswahl(dbsock,argc,argv)`: listet Signaldateien auf mit dem gesuchten Dateistamm.
- `ab_file_datum_auswahl(dbsock,argc,argv)`: listet Signaldateien auf mit dem gesuchten Datum.
- `ab_file_endung_auswahl(dbsock,argc,argv)`: listet Signaldateien auf mit der gesuchten Dateierstreckung.
- `ab_file_quelle_auswahl(dbsock,argc,argv)`: listet Signaldateien auf mit der gesuchten Quelle.
- `ab_file_pfad_auswahl(dbsock,argc,argv)`: listet Signaldateien auf mit dem gesuchten Pfad.

Untermenüpunkt Dateianzahl

1:	Dateianzahl-Allgemein	ab_file_anzahl(dbsock,int)
2:	Dateianzahl-Stamm	ab_file_anzahl_stamm(dbsock,argc,argv)
3:	Dateianzahl-Datum	ab_file_anzahl_datum(dbsock,argc,argv)
4:	Dateianzahl-Uhrzeit	ab_file_anzahl_uhrzeit(dbsock,argc,argv)
5:	Dateianzahl-Extension	ab_file_anzahl_endung(dbsock,argc,argv)
6:	Dateianzahl-Quelle	ab_file_anzahl_quelle(dbsock,argc,argv)
0:	Zurück	

Wobei:

- `ab_file_anzahl(dbsock,int)`: liefert die Anzahl aller Signaldateien in der Datenbank zurück.
- `ab_file_anzahl_stamm(dbsock,argc,argv)`: liefert die Anzahl aller Signaldateien mit dem entsprechenden Dateistamm zurück.
- `ab_file_anzahl_datum(dbsock,argc,argv)`: liefert die Anzahl aller Signaldateien mit dem entsprechenden Datum zurück.
- `ab_file_anzahl_uhrzeit(dbsock,argc,argv)`: liefert die Anzahl aller Signaldateien mit der entsprechenden Uhrzeit zurück.
- `ab_file_anzahl_endung(dbsock,argc,argv)`: liefert die Anzahl aller Signaldateien mit der entsprechenden Quelle zurück.

Untermenüpunkt Prozedur-Abfragen

1:	Alle Prozedure	ab_prozedur_alle(dbsock,int)
2:	Prozedur-Datei-Zuordnung	ab_prozedur_zugeordnet(dbsock,int)
3:	Prozedur-Dateiextension-Zuordnung	ab_fileext_prozedur_zu(dbsock,argc,argv)
4:	Prozedurauswahl nach Namen	ab_prozedur_name_auswahl(dbsock,argc,argv)
5:	Prozedurauswahl nach Typen	ab_prozedur_typ_auswahl(dbsock,argc,argv)
0:	Zurück	

Wobei:

- `ab_prozedur_alle(dbsock,int)`: listet alle Prozeduren in der Datenbank auf.
- `ab_prozedur_zugeordnet(dbsock,int)`: listet diejenige Prozeduren auf, die einer Signaldatei zugeordnet sind. Eine Prozedur ist einer Signaldatei zugeordnet, wenn der Identifikator der Prozedur in der Tabelle FILE_PRO_ZUORDNUNG im Feld FLP_ZU_ID zu finden ist.
- `ab_fileext_prozedur_zu(dbsock,argc,argv)`: listet alle Zuordnungen zwischen Dateiextensionen und Prozeduren in der Tabelle FILEEXT_PROZEDUR_ZU auf.
- `ab_prozedur_name_auswahl(dbsock,argc,argv)`: listet Prozeduren mit dem entsprechenden Namen auf.
- `ab_prozedur_typ_auswahl(dbsock,argc,argv)`: listet Prozeduren mit dem entsprechenden Typ auf.

Untermenüpunkt Format-Abfragen

1:	Alle Formate	ab_format_alle(dbsock,int)
2:	Zugeordnete Formate	ab_format_zugeordnet(dbsock,int)
3:	Formatauswahl nach Kennzeichnung	ab_format_name_auswahl(dbsock,argc,argv)
0:	Zurück	

Wobei:

- `ab_format_alle(dbsock,int)`: listet alle vorkommenden Formate in der Datenbank auf.
- `ab_format_zugeordnet(dbsock,int)`: listet diejenige Formate auf, die einer Signaldatei zugeordnet sind. Die Zuordnung zwischen Formaten und Signaldateien wird in der Tabelle `FILE_FO_ZUORDNUNG` festgehalten, wobei der Identifikator des zugeordneten Formats im Feld `FILE_FO_ZU_ID` zu finden ist.
- `ab_format_name_auswahl(dbsock,argc,argv)`: listet Formate mit der entsprechenden Kennzeichnung auf.

Untermenüpunkt Haupttyp-Abfragen

1:	Alle Haupttypen	ab_haupttyp_alle(dbsock,int)
2:	Zugeordnete Haupttypen	ab_haupttyp_zugeordnet(dbsock,int)
3:	Haupttypauswahl nach Kennzeichnung	ab_haupttyp_name_auswahl(dbsock,argc,argv)
0:	Zurück	

Wobei:

- `ab_haupttyp_alle(dbsock,int)`: listet alle vorkommenden Haupttypen in der Datenbank auf.
- `ab_haupttyp_zugeordnet(dbsock,int)`: listet diejenige Haupttypen auf, die einer Signaldatei zugeordnet sind. Die Zuordnung zwischen Haupttypen und Signaldateien wird in der Tabelle `FILE_HT_ZUORDNUNG` festgehalten, wobei der Identifikator des zugeordneten Haupttyps im Feld `FILE_HT_ZU_ID` zu finden ist.
- `ab_haupttyp_name_auswahl(dbsock,argc,argv)`: listet Haupttypen mit der entsprechenden Kennzeichnung auf.

Untermenüpunkt Untertyp-Abfragen

1:	Alle Untertypen	ab_untertyp_alle(dbsock,int)
2:	Zugeordnete Untertypen	ab_untertyp_zugeordnet(dbsock,int)
3:	Untertypauswahl nach Kennzeichnung	ab_untertyp_name_auswahl(dbsock,argc,argv)
0:	Zurück	

Wobei:

- `ab_untertyp_alle(dbsock,int)`: listet alle vorkommenden Untertypen in der Datenbank auf.
- `ab_untertyp_zugeordnet(dbsock,int)`: listet diejenige Untertypen auf, die einer Signaldatei zugeordnet sind. Die Zuordnung zwischen Untertypen und Signaldateien wird in der Tabelle `FILE_UT_ZUORDNUNG` festgehalten, wobei der Identifikator des zugeordneten Untertyps im Feld `FILE_UT_ZU_ID` zu finden ist.
- `ab_untertyp_name_auswahl(dbsock,argc,argv)`: listet Untertypen mit der entsprechenden Kennzeichnung auf.

Untermenüpunkt Quelle–Abfragen

1:	Alle Quellen	ab_quelle_alle(dbsock,int)
2:	Zugeordnete Quellen	ab_quelle_zugeordnet(dbsock,int)
3:	Quelle–Auswahl nach Namen	ab_quelle_name_auswahl(dbsock,argc,argv)
0:	Zurück	

Wobei:

- ab_quelle_alle(dbsock,int): listet alle vorkommenden Quellen in der Datenbank auf.
- ab_quelle_zugeordnet(dbsock,int): listet diejenige Quellen auf, die einer Signaldatei zugeordnet sind. Die Zuordnung zwischen Quellen und Signaldateien wird in der Tabelle FILE_QU_ZUORDNUNG festgehalten, wobei der Identifikator der zugeordneten Quelle im Feld FI_QU_ZU_ID zu finden ist.
- ab_quelle_name_auswahl(dbsock,argc,argv): listet Quellen mit dem entsprechenden Namen auf.

Untermenüpunkt Zugriffsrecht–Abfragen

1:	Alle Zugriffsrechte	ab_zugriff_alle(dbsock,int)
2:	Zugeordnete Zugriffsrechte mit Dateien	ab_zugriff_zugeordnet(dbsock,int)
3:	Zurück	

Wobei:

- ab_zugriff_alle(dbsock,int): listet alle Zugriffsberechtigungen in der Datenbank auf.
- ab_zugriff_zugeordnet(dbsock,int): listet diejenige Zugriffsrechte auf, die einer Signaldatei zugeordnet sind. Die Zuordnung zwischen Zugriffsrechten und Signaldateien wird in der Tabelle FILE_ZUG_ZUORDNUNG festgehalten, wobei der Identifikator der zugeordneten Quelle im Feld FI_ZUG_ZU_ID zu finden ist.

Hauptmenüpunkt Datei–Umwandeln

Wenn dieser Menüpunkt ausgewählt wird, wird die Funktion ab_file_umwandeln() sofort ausgeführt. Dieser Funktion wird der Datenbankhandle "dbsock" als Parameter übergeben.

Hauptmenüpunkt Datenbankroutinen–Eintragen

Dieser Menüpunkt führt zu folgenden Untermenüpunkten:

1:	Datei eintragen	Untermenü
2:	Dateiextension–Prozedur–Zuordnung eintragen	db_fileext_prozedur_zuordnen(dbsock,argc,argv)
3:	Prozedur eintragen	db_prozedur_eintragen(dbsock,argc,argv)
4:	Datei–Prozedur zuordnen	db_file_prozedur_zuordnen(dbsock,argc,argv)
5:	Format eintragen	db_format_eintragen(dbsock,argc,argv)
6:	Datei–Format zuordnen	db_file_format_zuordnen(dbsock,argc,argv)
7:	Haupttyp eintragen	db_haupttyp_eintragen(dbsock,argc,argv)
8:	Datei–Haupttyp zuordnen	db_file_haupttyp_zuordnen(dbsock,argc,argv)
9:	Untertyp eintragen	db_untertyp_eintragen(dbsock,argc,argv)
10:	Datei–Untertyp zuordnen	db_file_untertyp_zuordnen(dbsock,argc,argv)
11:	Datei–Zugriffsrecht zuordnen	db_file_zugriff_zuordnen(dbsock,argc,argv)
12:	Quelle eintragen	db_quelle_eintragen(dbsock,argc,argv)
13:	Datei–Quelle zuordnen	db_file_quelle_zuordnen(dbsock,argc,argv)
0:	Zurück	

Untermenüpunkt Datei eintragen

1:	Interaktiv	db_file_eintragen(dbsock,argc,argv)
2:	Einlesen von Dateien	db_file_einlesen_haupt(dbsock,"r",db_file_read_file,argc,argv)
0:	Zurück	

Wobei:

- db_file_eintragen(dbsock,argc,argv) dient der manuellen Eingabe von Signaldaten.
- db_file_einlesen_haupt() dient sowohl der Eingabe von Signaldaten von einer separaten Inputdatei als auch dem Löschen der Signaldaten gesteuert von einer Inputdatei. Diese Funktion wird wie folgt definiert: `db_file_einlesen_haupt(int dbsock,char* op,int(*) (FILE*,int,int,char**,char*),argc,argv)`.

Wobei:

1.Parameter	Datenbankhandle
2.Parameter	Operation Einlesen (r) oder Löschen (d)
3.Parameter	Zeiger auf eine weitere Funktion, die die eigentliche Operation durchführt

In diesem Fall ist der 3.Parameter die Funktion db_file_read_file zum Einlesen von Signaldaten in die Tabelle FILE, welche folgendermaßen definiert ist: `db_file_read_file(FILE*,int,int,char**,char*)`.

Hauptmenüpunkt Datenbankroutinen-Löschen

Dieser Menüpunkt führt zu folgenden Untermenüpunkten:

1:	Datei löschen-Interaktiv	db_file_löschen(dbsock,argc,argv)
2:	Datei löschen-Batch	db_file_einlesen_haupt(dbsock,"d",db_file_delete_file,argc,argv)
3:	Format löschen	db_format_löschen(dbsock,argc,argv)
4:	Prozedur löschen	db_prozedur_löschen(dbsock,argc,argv)
5:	Haupttyp löschen	db_haupttyp_löschen(dbsock,argc,argv)
6:	Untertyp löschen	db_untertyp_löschen(dbsock,argc,argv)
7:	Quelle löschen	db_quelle_löschen(dbsock,argc,argv)
8:	Dateizuordnung löschen	Untermenü
9:	Formatzuordnung löschen	db_format_zuordnung_löschen(dbsock,argc,argv)
10:	Prozedurzuordnung löschen	db_prozedur_zuordnung_löschen(dbsock,argc,argv)
11:	Prozedur-Dateiextension-Zuordnung löschen	db_fileext_prozedur_zu_löschen(dbsock,argc,argv)
12:	Prozedur-Dateiextension-Zuordnung Ziel löschen	db_fileext_prozedur_zu_löschen_ziel(dbsock,argc,argv)
13:	Haupttypzuordnung löschen	db_haupttyp_zuordnung_löschen(dbsock,argc,argv)
14:	Untertypzuordnung löschen	db_untertyp_zuordnung_löschen(dbsock,argc,argv)
15:	Quellezuordnung löschen	db_quelle_zuordnung_löschen(dbsock,argc,argv)
16:	Zugriffszuordnung löschen	db_zugriff_zuordnung_löschen(dbsock,argc,argv)
17:	Tabelle komplett löschen	db_tabelle_allgemein(dbsock,argc,argv)
0:	Zurück	

Unter Menüpunkt "Datei löschen-Batch" wird wiederum die Funktion db_file_einlesen_haupt() herangezogen. Im Gegensatz zum Untermenüpunkt "Datei eintragen-Einlesen von Dateien" wird in diesem Fall die Funktion db_file_delete_file als 3.Parameter übergeben. Die Funktion db_file_delete_file ist verantwortlich für das von einer Inputdatei gesteuerte Löschen von Signaldaten und wird in Bezug auf die Parameter genau so definiert wie die Funktion db_file_read_file().

3.3.2 Untermenüpunkt Dateizuordnung löschen

1:	Datei-Format-Zuordnung löschen	db_file_format_zuordnung_loeschen(dbsock,argc,argv)
2:	Datei-Prozedur-Zuordnung löschen	db_file_prozedur_zuordnung_loeschen(dbsock,argc,argv)
3:	Datei-Haupttyp-Zuordnung löschen	db_file_haupttyp_zuordnung_loeschen(dbsock,argc,argv)
4:	Datei-Untertyp-Zuordnung löschen	db_file_untertyp_zuordnung_loeschen(dbsock,argc,argv)
5:	Datei-Quelle-Zuordnung löschen	db_file_quelle_zuordnung_loeschen(dbsock,argc,argv)
6:	Datei-Zugriff-Zuordnung löschen	db_file_zugriff_zuordnung_loeschen(dbsock,argc,argv)
0:	Zurück	

Hauptmenüpunkt Datenbankroutinen-Änderungen

Dieser Menüpunkt führt zu folgenden Untermenüpunkten:

1:	Änderungen-Date	Untermenü
2:	Änderungen-Format	db_change_format(dbsock,argc,argv)
3:	Änderungen-Prozedur	Untermenü
4:	Änderungen-Haupttyp	db_change_haupttyp(dbsock,argc,argv)
5:	Änderungen-Untertyp	db_change_untertyp(dbsock,argc,argv)
6:	Änderungen-Quelle	db_change_quelle(dbsock,argc,argv)
7:	Änderungen-Datei-Zuordnung	Untermenü
8:	Änderungen-Format-Zuordnung	db_change_format_zuordnung(dbsock,argc,argv)
9:	Änderungen-Prozedur-Zuordnung	db_change_prozedur_zuordnung(dbsock,argc,argv)
10:	Änderungen-Haupttyp-Zuordnung	db_change_haupttyp_zuordnung(dbsock,argc,argv)
11:	Änderungen-Untertyp-Zuordnung	db_change_untertyp_zuordnung(dbsock,argc,argv)
12:	Änderungen-Quelle-Zuordnung	db_change_quelle_zuordnung(dbsock,argc,argv)
13:	Änderungen-Prozedur-Dateiextension-Zuordnung	Untermenü
14:	Änderungen-Zugriff-Zuordnung	db_change_zugriff_zuordnung(dbsock,argc,argv)
0:	Zurück	

Untermenü Änderungen-Datei

1:	Datei-Änderungen-Allgemein	db_change_file(dbsock,argc,argv)
2:	Datei-Änderungen-Namen	db_change_file_namen(dbsock,argc,argv)
3:	Datei-Änderungen-Stamm	db_change_file_stamm(dbsock,argc,argv)
4:	Datei-Änderungen-Datum	db_change_file_datum(dbsock,argc,argv)
5:	Datei-Änderungen-Uhrzeit	db_change_file_uhrzeit(dbsock,argc,argv)
6:	Datei-Änderungen-Extension	db_change_file_endung(dbsock,argc,argv)
7:	Datei-Änderungen-Pfad	db_change_file_pfad(dbsock,argc,argv)
8:	Datei-Änderungen-Quelle	db_change_file_quelle(dbsock,argc,argv)
0:	Zurück	

3.3.3 Untermenü Änderungen-Prozedur

1:	Prozedur-Änderungen-Allgemein	db_change_prozedur(dbsock,argc,argv)
2:	Prozedur-Änderungen-Namen	db_change_prozedur_namen(dbsock,argc,argv)
3:	Prozedur-Änderungen-Typ	db_change_prozedur_typ(dbsock,argc,argv)
0:	Zurück	

Änderungen-Datei-Zuordnung

1:	Dateizuordnungen-Änderungen-Allgemein	db_change_file_zuordnung(dbsock,argc,argv)
2:	Datei-Format-Zuordnung-Änderungen	db_change_file_format_zuordnung(dbsock,argc,argv)
3:	Datei-Prozedur-Zuordnung-Änderungen	db_change_file_prozedur_zuordnung(dbsock,argc,argv)
4:	Datei-Haupttyp-Zuordnung-Änderungen	db_change_file_haupttyp_zuordnung(dbsock,argc,argv)
5:	Datei-Untertyp-Zuordnung-Änderungen	db_change_file_untertyp_zuordnung(dbsock,argc,argv)
6:	Datei-Quelle-Zuordnung-Änderungen	db_change_file_quelle_zuordnung(dbsock,argc,argv)
7:	Datei-Zugriff-Zuordnung-Änderungen	db_change_file_zugriff_zuordnung(dbsock,argc,argv)
0:	Zurück	

Untermenüpunkt Änderungen-Prozedur-Dateiextension

1:	Prozedur-Dateiextension-Zuordnung-Prozedurnamen	db_change_fileext_prozedur_zu_pro(dbsock,argc,argv)
2:	Prozedur-Dateiextension-Zuordnung-Dateiextension	db_change_fileext_prozedur_zu_ext(dbsock,argc,argv)
3:	Prozedur-Dateiextension-Zuordnung-Zieldateiextension	db_change_fileext_prozedur_zu_zielext(dbsock,argc,argv)
0:	Zurück	

Hauptmenüpunkt Beenden

Mit diesem Menüpunkt wird das Verwaltungsprogramm beendet.

3.4 Funktionalitäten

In den vorhergehenden Abschnitten bekommt man eine kurze Übersicht über das gesamte Menüsystem des C-Programms angezeigt. In diesem Abschnitt wird näher erläutert, was der einzelne Menüpunkt macht, welche Inputwerte bzw. Argumente benötigt sind, falls eine Routine aufgerufen wird, und wie die entsprechende Kommandozeileoption lautet.

Vom modulartigen Aufbau und vom Aufbau des Menüsystems her ist es ersichtlich, daß sich diese Aufgliederung nach funktionalen Kategorien richtet. Wenn man allerdings von den zwei Bereichen "Datenbankverwaltung" und "Sprachsignal" ausgeht, gibt es dann prinzipiell zwei Hauptkategorien von implementierten Routinen:

- Routinen zur Datenmanipulation (D).

Routinen dieser Kategorie simulieren das funktionale Verhalten einiger mitgelieferten Datenbanktools von mSQL (in erster Linie msq). Auf solche Weise kann der Benutzer unmittelbar vom Anwendungsprogramm ausgehend Daten bearbeiten, ohne daß die entsprechenden Tools separat gestartet werden müssen. Außerdem sind die Routinen speziell auf anwendungsspezifische Gegebenheiten gemünzt und aus Benutzersicht nicht so sehr sperrig wie beim Eintippen von SQL-Statements mit dem Tool msq. Darüber hinaus erfordert die Arbeit mit dem Tool "msq" gewisse Grundkenntnisse im Datenbankbereich (nämlich in der Formulierung von SQL-Statements).

- Routinen zur Sprachsignal-Umwandlung (S).

Routinen dieser Kategorie sind anwendungsspezifisch und sind bei der Formatumwandlung erforderlich.

Im folgenden werden der Reihe nach alle vorkommenden Menüpunkte unter der Lupe genommen. Es gibt sicher an der einen oder der anderen Stelle Überschneidungen mit der Beschreibung im vorangehenden Abschnitt. Trotzdem ist es sinnvoll, eine Art detaillierte Beschreibung der einzelnen Menüpunkte darzubieten. Es ist dabei aber zu beachten, daß in diesem Abschnitt jedes einzelne Menü im Vergleich zum letzten Abschnitt anders benannt wird, und zwar den jeweiligen Kommandozeileoptionen entsprechend. Somit wird auch ersichtlich, wie ein Menüpunkt durch Kommandozeileoption aufgerufen wird. In Bezug auf die zwei Startmodi (partielle Atomatisierung und vollständige Automatisierung) handelt es sich hierbei um die zugrundeliegende Ablauflogik des Verwaltungsprogramms, die ja das gesamte Konstrukt für die zwei Startmodi ist. Dementsprechend wird in diesem Abschnitt die Befehlkette im vollständigen Automatisierungs-Startmodus ebenfalls erläutert und zusammengefaßt, sofern eine mit dem angesprungenen Menüpunkt assoziiert ist und gestartet werden soll.

3.4.1 Menü A

- **1: bf Abfragen**

Dieser Menüpunkt führt zu weiteren Menüpunkten, mit denen Anfragen an den Datenbankserver übertragen werden können.

Option A=1
Menüsystem
Kategorie D

- **2: Datei umwandeln**

Mit diesem Menüpunkt kann der Benutzer Formatumwandlungen durchführen.

Option A=2
Menüsystem
Kategorie S

- **3: Datenbankroutinen: Eintragen**

Dieser Menüpunkt führt zu Routinen, die zum Eintragen von Daten in die Tabellen behilflich sind.

Option A=3
Menüsystem
Kategorie D

- **4: Datenbankroutinen: Löschen**

Dieser Menüpunkt führt zu Routinen, die zum Löschen von Daten in den Tabellen verwendet werden.

Option A=4
Menüsystem
Kategorie D

- **5: Datenbankroutinen: Änderungen**

Dieser Menüpunkt führt zu Routinen, die zum Ändern von Daten in den Tabellen verwendet werden.

Option A=5
Menüsystem
Kategorie D

- **0: Beenden**

Dieser Menüpunkt führt zu Routinen, die zum Löschen von Daten verwendet werden.
Option A=0

3.4.2 Menü A1_B

Dieses Menü wird erreicht nach Eingabe der Kommandozeileoption A=1.

- **1: Datei-Abfragen**

Mit Hilfe dieses Menüpunkts kann der Benutzer dateibezogene Abfragen ausführen lassen.

Option A=1B=1

Menüsystem

- **2: Prozedur-Abfragen**

Mit Hilfe dieses Menüpunkts kann der Benutzer Abfrage starten, die auf Daten der Tabelle Prozedur zugreifen.

Option A=1B=2

Menüsystem

- **3: Format-Abfragen**

Mit diesem Menüpunkt können formatbezogene Abfragen an den Datenbankserver übergeben werden.

Option A=1B=3

Menüsystem

- **4: Haupttyp-Abfragen**

Dieser Menüpunkt ermöglicht, Abfragen zu formulieren, die auf Daten der Tabelle Haupttyp zugreifen.

Option A=1B=4

Menüsystem

- **5: Untertyp-Abfragen**

Dieser Menüpunkt ermöglicht, Abfragen zu formulieren, die auf Daten der Tabelle Untertyp zugreifen.

Option A=1B=5

Menüsystem

- **6: Quelle-Abfragen**

Dieser Menüpunkt ermöglicht, Abfragen zu formulieren, die auf Daten der Tabelle Quelle zugreifen. Option A=1B=6

Menüsystem

- **7: Zugriffsrecht-Abfragen**

Dieser Menüpunkt ermöglicht, Abfragen zu formulieren, die auf Daten der Tabelle Zugriffsrecht zugreifen.

Option A=1B=7

3.4.3 Menü A2_B

Bei Kommandozeileoption A=2 wird zu diesem Menü gesprungen.

- **1: Umwandlung Alle**

Dieser Menüpunkt ist mit der Routine `ab_file_umwandeln_haupt` assoziiert, die unter anderem zwei Parameter besitzt: den bekannten Socket-Deskriptor und eine Zeichenkette mit spezifischer Bedeutung. In diesem Fall ist der zweite Parameter eine Leerzeichenkette. Nach Auswahl dieses Menüpunkts wird der Benutzer aufgefordert, den Namen einer Inputdatei einzugeben, die dermaßen gestaltet ist:

Die Inputdatei enthält Datenzeilen, die jeweils den Namen einer Sprachsignaldatei enthalten. Eine Beschreibung des Formats der Inputdatei findet sich im Abschnitt 5.1.4 wieder.

Danach wird der Inhalt der Inputdatei Zeile für Zeile abgetastet und die darin enthaltenen Daten der Routine `ab_file_umwandeln_mname` übergeben. Die Routine `ab_file_umwandeln_mname` schaut zuerst in der Tabelle `FILE` nach, ob der übergebene Dateiname dort zu finden ist. Im positiven Fall wird anhand der Tabelle `file_ext_prozedur_zu` diejenige Prozeduren der Reihe nach aufgerufen, die mit der Dateiextension der aktuellen vorliegenden Datei assoziiert sind. An dieser Stelle ist es sinnvoll in Erinnerung zu rufen, daß eine Reihe von Prozeduren einer einzigen Dateiextension zugeordnet werden können. Der Aufruf der Routine `ab_file_umwandeln_mnamen` wiederholt sich für jede Datenzeile, bis die Inputdatei abgearbeitet ist. Das Ergebnis der Formatumwandlung ist im Normalfall eine Outputdatei, die in Abhängigkeit von der Konfigurationseinstellung der Transformationsprozeduren und dem in der Globalvariablen "ExPho" definierten Wert in einem bestimmten Verzeichnis gespeichert wird (entweder aktuelles Verzeichnis, wo das Verwaltungsprogramm aufgerufen wird, oder im durch "ExPho" spezifizierten Verzeichnis - siehe auch 3.2.3 und 7.3).

Wenn der Benutzer `< RETURN >` drückt, ohne daß der Name der Inputdatei eingegeben wird, wird die Routine `ab_file_umwandeln` gestartet. Anschließend kommt die Aufforderung, den Namen der umzuwandelnden Datei einzugeben. Falls keinerlei Eingabe getätigt wird, wird die Routine beendet und es erscheint wieder das Menüsystem.

Die Extension der Ergebnisdatei wird ebenfalls der Tabelle `fileext_prozedur_zu` entnommen. Überdies wird die Ergebnisdatei sowohl in die Tabelle `FILE` als auch in eine von den Tabellen `FILE_SPH`, `FILE_PHONES`, `FILE_WORDS` eingetragen, sofern deren Namen darin noch nicht vermerkt ist.

Option A=2B=1

Zusammenfassung:

Eingabe: Inputdatei oder
Dateinamen (obligatorisch)

Ausgabe: Resultatdateien, deren Dateiextension gemäß der Zuordnung in der Tabelle `fileext_prozedur_zu` bestimmt wird

Automatisierungsmodus:

```
korpo -s "Datenbankname" A=2 B=1 -b "Inputdatei" -e
oder
korpo -s "Datenbankname" A=2 B=1 -f "Dateinamen" -e
```

- **2: Umwandlung-Sph**

Im Vergleich zum Menüpunkt A=2B=1 soll hier nur die Formatumwandlung ins Sph-Format für Dateien mit Sd-Endung durchgeführt werden. Dazu ist das Programm "e2sphere" prädestiniert. D.h., es wird nicht in der Tabelle fileext_prozedur_zu nach allen assoziierten Prozeduren gesucht. Auch die Routine ab_file_umwandeln_haupt wird aufgerufen, der zweite Parameter ist aber nun nicht leer, sondern beinhaltet die Zeichenkette "Sph". Am Anfang wird der Benutzer auch aufgefordert, eine Inputdatei anzugeben, welche die umzuwandelnden Dateien enthält und dieselbe Struktur hat wie die im Fall "Umwandlung Alle" verwendeten Inputdatei. Falls eine Inputdatei eingegeben ist, werden all die darin enthaltenen Dateien ins Sph-Format mit Hilfe der Routine ab_file_umwandeln_mnamen umgewandelt. Der Parameter "Sph" wird dabei an die Routine ab_file_umwandeln_mnamen durchgereicht, so daß die Dateierweiterung von vorne herein als "Sph" festgelegt wird.

Im negativen Fall (keine Dateieingabe) wird die Routine ab_file_umwandeln_sph gestartet, die sich der Routine e2sphere bedient und die Formatumwandlung nach Sph-Format für eine Sprachsignaldatei durchführt. Auch hier wird der Benutzer aufgefordert, entweder den Namen einer Sprachsignaldatei oder ein Datum und eine Dateierweiterung einzugeben, um die Existenz der umzuwandelnden Datei in der Datenbank zu überprüfen. Der Ort der Ergebnisdatei wird ebenso wie oben beschrieben bestimmt.

Option A=2B=2

Zusammenfassung:

Eingabe: Inputdatei oder

Dateinamen (obligatorisch)

Ausgabe: Resultatdateien mit der Dateierweiterung "Sph"

Automatisierungsmodus:

korpo -s "Datenbankname" A=2 B=2 -b "Inputdatei" -e

oder

korpo -s "Datenbankname" A=2 B=2 -f "Dateinamen" -e

- **3: Umwandlung-Phones**

Dieser Menüpunkt ist zuständig für die Formatumwandlung nach Phones-Format. Dabei wird der Routine ab_file_umwandeln_haupt der Parameter "Phones" übermittelt. Auch hier kommt die Aufforderung ins Spiel, eine Inputdatei einzugeben. Im positiven Fall wird die Routine ab_file_umwandeln_mnamen mit dem übergebenen Parameter "Phones" aktiviert, die die Umwandlung ausführt, bis die Inputdatei abgearbeitet wird. Wenn keine Inputdatei angegeben ist, wird die entsprechende Routine namens ab_file_umwandeln_phones aufgerufen. In beiden Fällen macht das vordefinierte Programm "Alignphones" die eigentliche Arbeit, die Resultatdatei hat automatisch die Extension "Phones".

Option A=2B=3

Zusammenfassung:

Eingabe: Inputdatei oder

Dateinamen (obligatorisch)

Ausgabe: Resultatdateien mit der Dateierweiterung "Phones"

Automatisierungsmodus:

```
korpo -s "Datenbankname" A=2 B=3 -b "Inputdatei" -e
oder
korpo -s "Datenbankname" A=2 B=3 -f "Dateinamen" -e
```

- **4: Umwandlung–Words**

Wie bei der Umwandlung nach Phones–Format kann man bei diesem Menüpunkt Sprachsignaldateien ins Words–Format überführen. In diesem Fall wird der bekannten Routine `ab_file_umwandeln_mnamen` der Parameter "words" übergeben. Dieser Menüpunkt zeigt auch einen ähnlichen Ablauf wie bei der Formatumwandlung nach "Phones". Eine Inputdatei ist ebenfalls dabei, nur die verwendete Routine ist anders und heißt in diesem Fall konkret `ab_file_umwandeln_words`. Die dabei eingesetzte Transformationsprozedur heißt in diesem Fall "Alignwords". Die Extension der Resultatdatei ist "Words".

Option A=2B=4

Zusammenfassung:

Eingabe: Inputdatei oder

Dateinamen (obligatorisch)

Ausgabe: Resultatdateien mit der Dateiextension "words"

Automatisierungsmodus:

```
korpo -s "Datenbankname" A=2 B=4 -b "Inputdatei" -e
oder
korpo -s "Datenbankname" A=2 B=4 -f "Dateinamen" -e
```

- **5: Umwandlung–Extension Alle**

Es werden bei diesem Menüpunkt alle für die Datei möglichen Umwandlungen durchgeführt. Im Vergleich zur allgemeinen Formatumwandlung spielt hier die Routine `ab_file_umwandeln_endung` die gleiche Rolle wie die Routine `ab_file_umwandeln_haupt` bei den anderen Menüpunkten. Die Routine `ab_file_umwandeln_endung` besitzt ebenfalls u.a. zwei Parameter, den Socket–Deskriptor und eine Zeichenkette als Kennzeichnungsflag. Bei diesem Menüpunkt ist die Zeichenkette eine leere. Zuerst muß eine Dateiextension eingegeben werden. Anhand der eingegeben Dateiextension werden in der Tabelle FILE nach denjenigen Dateien mit der entsprechenden Extension gesucht. Danach wird eine Durchlaufschleife gebildet, in der die Routine `ab_file_umwandeln_endung_check` zusammen mit den extrahierten Daten der gefundenen Dateien aufgerufen wird. An die Routine `ab_file_umwandeln_endung_check` wird der Kennzeichnungsflag (hier leer) weitergereicht.

Die Routine `ab_file_umwandeln_endung_check` überprüft in der Tabelle FILE-EXT_PROZEDUR_ZU zuerst, ob Prozedure existieren, welche auf Dateien der ausgewählten Extension angewandt werden können. Dabei werden auch Informationen über die Extension der jeweiligen Resultatdatei ermittelt. Die Resultatdatei mit der richtigen Extension wird sowohl in die Tabelle FILE als auch in die entsprechende Protoll– Tabelle (FILE_SPH, FILE_PHONES, FILE_WORDS) eingetragen, sofern sie noch nicht drin ist. Falls die Tabelle FILEEXT_PROZEDUR_ZU leer ist oder keine entsprechende Zuordnung existiert, erfolgt ein Hinweis darauf und die Formatumwandlung findet nicht statt. Es muß an dieser Stelle hervorgehoben werden, daß

sich alle in der Tabelle FILE enthaltenen Dateien, deren Extension die angegebene Extension haben der Formatumwandlung unterziehen. Dieser Vorgang kann sehr zeitintensiv sein, wenn eine Vielzahl von Sprachsignaldateien qualifiziert sind. Aus diesem Grund soll dieser Umwandlungsvorgang nur in Batchform gestartet werden (beispielsweise übers Wochenende).

Option A=2B=5

Zusammenfassung:

Eingabe: Dateiextension (obligatorisch)

Ausgabe: Resultatdateien, deren Dateiextension gemäß der Zuordnung in der Tabelle fileext_prozedur_zu bestimmt wird

Automatisierungsmodus:

korpo -s "Datenbankname" A=2 B=5 "Dateiextension" -e

• **6: Umwandlung-Extension-sph**

Im Gegensatz zur allgemeinen, sich nach der Dateiextension richtenden Formatumwandlung kann in diesem Fall nur die Formatumwandlung nach Sph-Format durchgeführt werden. Der Routine ab_file_umwandeln_endung wird der Parameter "Sph" übergeben, der dann weiter an die Routine ab_file_umwandeln_endung_check durchgereicht wird. Der Ablauf verläuft sehr ähnlich wie bei der allgemeinen Formatumwandlung bezogen auf Dateiextensionen (Umwandlung-Extension Alle). Auch hier werden alle qualifizierten Dateien in der Tabelle FILE berücksichtigt.

Der Unterschied besteht in den folgenden Punkten:

- die Umwandlungsprozedur ist "e2sphere".
- Resultatdatei wird außer in die Tabelle FILE nur in die Tabelle FILE_SPH eingetragen.
- Resultatdatei besitzt in diesem Fall auf logische Weise die Extension "Sph".
- Es wird nicht in der Tabelle FILEEXT_PROZEDUR_ZU nachgeschaut, ob eine Zuordnung vorhanden ist.
- Nur Dateiextension **Sd** kann angegeben werden.

Option A=2B=6

Zusammenfassung:

Eingabe: Dateiextension (obligatorisch)

Ausgabe: Resultatdateien mit der Dateiextension "Sph"

Automatisierungsmodus:

korpo -s "Datenbankname" A=2 B=6 "Dateiextension" -e

• **7: Umwandlung-Extension-phones**

Dieser Menüpunkt ist prädestiniert für die Überführung von Sprachsignaldateien ins Phones-Format. Der Programmverlauf zeigt ein ähnliches Profil wie bei der Sph-Formatumwandlung (siehe oben).

Folgende Merkmale kennzeichnen den Programmverlauf dieses Menüpunkts:

- Die Umwandlungsprozedur ist "Alignphones".

- Resultatdatei wird außer in die Tabelle FILE nur in die Tabelle FILE_PHONES eingetragen.
- Resultatdatei besitzt in diesem Fall auf logische Weise die Extension "Phones".
- Es wird nicht in der Tabelle FILEEXT_PROZEDUR_ZU nachgeschaut, ob eine Zuordnung vorhanden ist.
- Nur Dateierweiterung **Sd** kann angegeben werden.

Option A=2B=7

Zusammenfassung:

Eingabe: Dateierweiterung (obligatorisch)

Ausgabe: Resultatdateien mit der Dateierweiterung "Phones"

Automatisierungsmodus:

korpo -s "Datenbankname" A=2 B=7 "Dateierweiterung" -e

• **8: Umwandlung-Extension-words**

Dieser Menüpunkt ist prädestiniert für die Überführung von Sprachsignaldateien ins Words-Format. Der Programmverlauf zeigt ein ähnliches Profil wie bei der Sph-Formatumwandlung (siehe oben).

Folgende Merkmale kennzeichnen den Programmverlauf dieses Menüpunkts:

- die Umwandlungsprozedur ist "Alignwords".
- Resultatdatei wird außer in die Tabelle FILE nur in die Tabelle FILE_WORDS eingetragen.
- Resultatdatei besitzt in diesem Fall auf logische Weise die Extension "words".
- Es wird nicht in der Tabelle FILEEXT_PROZEDUR_ZU nachgeschaut, ob eine Zuordnung vorhanden ist.
- Nur Dateierweiterung **Sd** kann angegeben werden.

Option A=2B=8

Zusammenfassung:

Eingabe: Dateierweiterung (obligatorisch)

Ausgabe: Resultatdateien mit der Dateierweiterung "Words"

Automatisierungsmodus:

korpo -s "Datenbankname" A=2 B=8 "Dateierweiterung" -e

3.4.4 Menü A3_B

• **1: Datei eintragen**

Dieser Menüpunkt ist der Einstieg zu einem anderen Menü auf einer tiefer gelegten Ebene, mit dem Daten in die Tabelle FILE eingetragen werden können. Das Eintragen von Daten geschieht auf zwei Wegen:

- interaktiv
- durch eine geeignete Inputdatei

Menüsystem

Option A=3B=1

- **2: Dateiextension-Prozedur-Zuordnung eintragen**

Mit diesem Menüpunkt kann die Zuordnung zwischen einer Dateiextension und denjenigen Programmen (Prozeduren) in die Tabelle FILEEXT_PROZEDUR_ZU eingetragen werden. Wie bereits erwähnt sind die zugeordneten Prozeduren diejenigen, die mit den Dateien mit der zugewiesenen Extension operieren können. Dieser Menüpunkt aktiviert seinerseits die Routine `db_fileext_prozedur_zuordnen`, die wie folgt verläuft:

Aufforderung, eine Dateiextension einzugeben.

Aufforderung, einen Prozedurnamen einzugeben.

Aufforderung, eine Dateiextension für die Zieldatei einzugeben.

Überprüfung, ob die eingegebene Prozedur in der Tabelle PROZEDUR vorhanden ist
 Überprüfung in der Tabelle FILE, ob eine Datei die eingegebene Extension hat
 Der entsprechende Eintrag wird in die Tabelle FILEEXT_PROZEDUR_ZU gemacht, sofern die zwei Prüfungen positive Ergebnisse liefern.

Option A=3B=2

Zusammenfassung:

Eingabe: Prozedurname, Dateiextension, Dateiextension der Zieldatei (alle obligatorisch)

Automatisierungsmodus:

```
korpo -s "Datenbankname" A=3 B=2 "Prozedurname" "Dateiextension"
"Zieldateiextension" -e
```

- **3: Prozedur eintragen**

Dieser Menüpunkt ist zuständig für die Eintragung von Prozeduren in die Tabelle PROZEDUR. Dabei wird die Routine `db_prozedur_eintragen` aufgerufen. Der Benutzer muß unbedingt den Prozedurnamen eingeben. Die Angabe über den Prozedurtyp ist optional. Danach wird überprüft, ob die Prozedur bereits in der Tabelle PROZEDUR zu finden ist. Im negativen Fall wird die neue Prozedur in die Datenbank eingetragen.

Option A=3B=3

Zusammenfassung:

Eingabe: Prozedurname (obligatorisch), Prozedurtyp (optional)

Automatisierungsmodus:

```
korpo -s "Datenbankname" A=3 B=3 "Prozedurname" "Prozedurtyp" -e
oder
korpo -s "Datenbankname" A=3 B=3 "Prozedurname" -e
```

- **4: Datei-Prozedur zuordnen**

Mit diesem Menüpunkt kann der Benutzer die Zuordnung zwischen einer Prozedur und einer Sprachsignaldatei herstellen. Dazu wird die Routine `db_file_prozedur_zuordnen`

aktiviert. Als Eingabe werden erwartet der Prozedurname und der Dateiname. Anschließend wird nachgeschaut, ob diese Zuordnung bereits besteht. Soll es nicht der Fall sein, wird die neue Zuordnung in die Tabelle FILE_PRO_ZUORDNUNG eingetragen.

Option A=3B=4

Zusammenfassung:

Eingabe: Prozedurname, Dateiname (alle obligatorisch)

Automatisierungsmodus:

korpo -s "Datenbankname" A=3 B=4 "Prozedurname" "Dateinamen"
"Zieldateiextension" -e

• **5: Format eintragen**

Die Bezeichnung dieses Menüpunkts läßt einen erraten, was man damit anstellen kann, nämlich, Formate in die Tabelle FORMAT einzutragen. Dafür ist die Routine db_format_eintragen zuständig. Als Eingabe wird die Formatkennzeichnung erwartet. Es wird sichergestellt, daß das neue Format in die Datenbank eingetragen wird, sofern es noch nicht in der Datenbank enthalten ist.

Option A=3B=5

Zusammenfassung:

Eingabe: Formatkennzeichnung (obligatorisch)

Automatisierungsmodus:

korpo -s "Datenbankname" A=3 B=5 "Formatkennzeichnung" -e

• **6: Datei-Format zuordnen**

Mit diesem Menüpunkt kann die Beziehung zwischen einem bestimmten Format und einer bestimmten Sprachsignaldatei erstellt werden. Dies stellt eine zusätzliche Informationsquelle dar, wenn der Name einer Sprachsignaldatei keinen Aufschluß über deren Format gibt.

In diesem Fall wird die Routine db_file_format_zuordnen herangezogen. Eingaben über Formatkennzeichnung und Dateinamen müssen unbedingt gemacht werden. Es wird auf jeden Fall nach Duplikaten gesucht, bevor die Eintragung der Daten in die Tabelle FILE_FO_ZUORDNUNG erfolgt.

Option A=3B=6

Zusammenfassung:

Eingabe: Formatkennzeichnung, Dateiname (alle obligatorisch)

Automatisierungsmodus:

korpo -s "Datenbankname" A=3 B=6 "Formatkennzeichnung" "Dateinamen" -e

- **7: Haupttyp eintragen**

Mit Hilfe dieses Menüpunkts kann der Benutzer neue Haupttypen in die Tabelle HAUPTTYP eintragen. Wie üblich wird dabei die Routine db_haupttyp_eintragen benötigt, die Angabe über Haupttyp-Kennzeichnung erwartet. Falls der neue Haupttyp in der Tabelle HAUPTTYP noch nicht vermerkt ist, wird er in die Datenbank aufgenommen.

Option A=3B=7

Zusammenfassung:

Eingabe: Haupttypkennzeichnung (obligatorisch)

Automatisierungsmodus:

korpo -s "Datenbankname" A=3 B=7 "Haupttypkennzeichnung" -e

- **8: Datei-Haupttyp zuordnen**

Die Beziehung zwischen Sprachsignaldateien und deren Haupttypen wird in der Tabelle FILE_HT_ZUORDNUNG ausgedrückt. Dieser Menüpunkt bietet die Möglichkeit an, neue Einträge in diese Tabelle zu tätigen. Dabei wird die Routine db_file_haupttyp_zuordnen aktiviert. Als Eingabe werden erwartet die Haupttypkennzeichnung und der Dateiname der entsprechenden Sprachsignaldatei. Es gilt wie immer, daß eine Duplikat-Überprüfung vor der Eintragung der Zuordnung in die Datenbank stattfindet.

Option A=3B=8

Zusammenfassung:

Eingabe: Haupttypkennzeichnung, Dateiname (alle obligatorisch)

Automatisierungsmodus:

korpo -s "Datenbankname" A=3 B=8 "Haupttypkennzeichnung" "Dateinamen" -e

- **9: Untertyp eintragen**

Dieser Menüpunkt dient der Eintragung von Untertyp-Informationen in die Tabelle UNTERTYP. Die Vorgehensweise ähnelt sich der bei der Eintragung von Hauttypen.

Option A=3B=9

Zusammenfassung:

Eingabe: Untertypkennzeichnung (obligatorisch)

Automatisierungsmodus:

korpo -s "Datenbankname" A=3 B=9 "Untertypkennzeichnung" -e

- **10:Datei-Untertyp zuordnen**

Analog zum Menüpunkt Datei-Haupttyp-Zuordnen werden hierbei entsprechende Daten in die Tabelle FILE_UT_ZUORDNUNG eingetragen. In diesem Fall übernimmt die Routine db_file_untertyp_zuordnen die Regie.

Option A=3B=10

Zusammenfassung:

Eingabe: Untertypkennzeichnung, Dateiname (alle obligatorisch)

Automatisierungsmodus:

korpo -s "Datenbankname" A=3 B=10 "Untertypkennzeichnung" "Dateinamen" -e

• **11:Datei-Zugriffsrecht zuordnen**

Mit diesem Menüpunkt können einer Sprachsignaldatei bestimmte Zugriffsrechte zugewiesen werden. Die einer Sprachsignaldatei zugewiesenen Rechte werden in der Tabelle FILE_ZUG_ZUORDNUNG gespeichert. Hierbei ist die Routine db_file_zugriff_zuordnen zuständig, die zwei Eingaben erwartet:Zugriffsrecht-ID und Dateinamen. Es ist an dieser Stelle sinnvoll, noch mal anzumerken, daß eine ID ein Zugriffsrecht repräsentiert und alle IDs von vorne herein definiert werden und in der Tabelle ZUGRIFFSRECHT untergebracht sind. In der Eintragungsroutine wird daher auch überprüft, ob die angegebene Zugriffsrecht-ID eine gültige ist oder nicht.

Option A=3B=11

Zusammenfassung:

Eingabe: Zugriffsrecht-ID, Dateinamen (alle obligatorisch)

Automatisierungsmodus:

korpo -s "Datenbankname" A=3 B=11 "Zugriffsrecht-ID" "Dateinamen" -e

• **12:Quelle eintragen**

Dieser Menüpunkt dient der Eintragung von entsprechenden Daten in die Tabelle QUELLE. Dazu wird die Routine db_quelle_eintragen verwendet. Nach Aufruf der Routine wird der Benutzer aufgefordert, zwei Bezeichnungen für die Quelle einzugeben, wobei die erste Bezeichnung obligatorisch und die zweite optional ist.

Option A=3B=12

Zusammenfassung:

Eingabe: Bezeichnung 1 (obligatorisch), Bezeichnung 2 (optional)

Automatisierungsmodus:

korpo -s "Datenbankname" A=3 B=12 "Bezeichnung 1" "Bezeichnung 2" -e
oder
korpo -s "Datenbankname" A=3 B=12 "Bezeichnung 1" -e

- **13:Datei-Quelle zuordnen**

Eine Sprachsignaldatei stammt irgendwie aus einer bestimmten Quelle. Diese kann eine Bezeichnung, ein Ort und ein Verzeichnis sein. Information darüber werden in der Tabelle FILE_QU_ZUORDNUNG festgehalten. Mit Hilfe dieses Menüpunkts können solche Informationen in die Datenbank übernommen werden. Dazu wird die Routine `db_file_quelle_zuordnen` herangezogen.

Option A=3B=13

Zusammenfassung:

Eingabe: Bezeichnung 1 der Quelle (obligatorisch), Bezeichnung 2 der Quelle (optional), Dateinamen (obligatorisch)

Automatisierungsmodus:

```
korpo -s "Datenbankname" A=3 B=13 "Bezeichnung 1" "Bezeichnung 2"
"Dateinamen" -e
```

oder

```
korpo -s "Datenbankname" A=3 B=13 "Bezeichnung 1" "Dateinamen" -e
```

3.4.5 Menü A4_B

- **1: Datei löschen-interaktiv**

Dieser Menüpunkt ist dazu da, um Daten von Sprachsignaldateien in der Tabelle FILE zu löschen. Dabei wird die Routine `db_file_löschen` verwendet, welche einen Dateinamen als obligatorische Eingabe erwartet. Anhand dieses Dateinamens kann die Routine in der Tabelle FILE nachschauen, ob sich die betroffene Datei darin befindet. Im positiven Fall wird sie von der Datenbank gelöscht. Um Daten immer möglichst konsistent zu halten, wird dieser Vorgang weiter propagiert, indem auch Dateneinträge mit der gelöschten Datei in den Tabellen FILE_PRO_ZUORDNUNG, FILE_FO_ZUORDNUNG, FILE_HT_ZUORDNUNG, FILE_UT_ZUORDNUNG, FILE_ZUG_ZUORDNUNG und FILE_QU_ZUORDNUNG mit gelöscht werden.

Soll der Name der zu löschenden Datei nicht eingegeben werden, sondern nur mit der RETURN-Taste quittiert, wird die Routine `db_file_löschen_allgemein()` aufgerufen. Innerhalb dieser Routine wird der Benutzer verlangt, Dateiextension, Dateipfad und Dateiquelle anzugeben. Allerdings sind all diese Angaben optional. Je nach Angabe wird eine entsprechende Anfrage an den Datenbankserver geschickt. Unter Umständen liefert die Anfrage eine Menge von Sprachsignaldateien aus der Tabelle FILE als Ergebnis zurück, die den Angaben gemäß gelöscht werden sollen. Aus diesem Grund erfolgt eine Rückfrage an den Benutzer, die bestätigt werden muß. Auf diese Weise können mehrere Dateien auf einmal gelöscht werden.

Option A=4B=1

Zusammenfassung:

Eingabe: Dateinamen i (obligatorisch)

oder

Dateiextension (optional)

Dateipfad (optional)
Dateiquelle (optional)

Automatisierungsmodus:

```
korpo -s "Datenbankname" A=4 B=1 "Dateinamen" -e
oder
korpo -s "Datenbankname" A=4 B=1 "Dateiextension" -e
oder
korpo -s "Datenbankname" A=4 B=1 "Dateipfad" -e
oder
korpo -s "Datenbankname" A=4 B=1 "Dateiquelle" -e
```

• 2: Datei löschen–Batch

Im Gegensatz zum Menüpunkt "Datei löschen – interaktiv" werden hier Dateien nicht durch interaktive Angaben gelöscht. Der Löschvorgang erfolgt viel mehr über eine Inputdatei, in der alle zu löschenden Sprachsignaldateien spezifiziert sind. An Stelle der Routine `db_file_löschen` tritt die vielfältige Routine `db_file_einlesen_haupt` in Erscheinung. Diese Routine ist vielfältig in dem Sinne, daß sie nicht nur in diesem Fall zu Hilfe genommen wird, sondern auch bei der Eintragung von Sprachsignaldateien in die Datenbank in Batch-Form (siehe 3.4.4). Um die Vorgehensweise dieser Routine verstehen zu können, soll die im folgenden näher betrachtet werden.

Die Routine `db_file_einlesen_haupt` besitzt u.a. 4 Parameter, die von wichtiger Bedeutung sind:

- den bekannten Socket-Deskriptor,
- den Namen der zu löschenden Datei,
- die Operationskennung: **d** für Löschen und **r** für Einlesen,
- den Handle auf eine weitere Routine, die situationsabhängig ist. D.h. beim Löschen zeigt dieser Handle auf die Routine `db_file_delete_file`, beim Einlesen zeigt er auf eine andere Routine, die später bei der Beschreibung des Einlesevorgangs noch mal zur Sprache kommt.

Wie bereits erwähnt enthält die Inputdatei außer den eigentlichen Daten auch eine Konfigurationszeile, die zur Interpretation der Daten dienen. Die Routine `db_file_einlesen_haupt` versucht zuerst, die Konfigurationsinformation von der Inputdatei auszulesen und kann auf diese Weise die Richtigkeit der Inputdatei bezogen auf deren Format bestimmen. Soll alles in Ordnung sein, hat die Routine genügend Informationen über die in der Inputdatei enthaltenen Daten gesammelt (beispielsweise Anzahl der Daten, Bedeutung einzelner Spalte). Diese gewonnenen Informationen werden zusammen mit weiteren Parametern wie dem Socket-Deskriptor und dem Namen der Inputdatei der Routine `db_file_delete_file` weitergereicht. Anhand dieser Konfigurationsinformationen werden innerhalb dieser Routine entsprechende Daten von der Inputdatei extrahiert und in die Tabelle FILE eingetragen.

Option A=4B=2

Zusammenfassung:

Eingabe: Namen der Inputdatei (obligatorisch)

Automatisierungsmodus:

```
korpo -s "Datenbankname" A=4 B=2 "Inputdatei" -e
```

- **3: Format löschen**

Mit Hilfe dieses Menüpunkts können Datensätze aus der Tabelle FORMAT gelöscht werden. Dabei wird die Routine `db_format_löschen` aufgerufen. Als Eingabe wird die Formatkennzeichnung benötigt. Die oben genannte Routine prüft zuerst, ob es das der Eingabe entsprechende Format in der Datenbank gibt. Im positiven Fall wird es gelöscht.

Option A=4B=3

Zusammenfassung:

Eingabe: Formatkennzeichnung (obligatorisch)

Automatisierungsmodus:

`korpo -s "Datenbankname" A=4 B=3 "Formatkennzeichnung" -e`

- **4: Prozedur löschen**

Dieser Menüpunkt dient dem Löschen von Datensätzen in der Tabelle PROZEDUR und bedient sich zu diesem Zweck der Routine `db_prozedur_löschen`. Diese Routine erwartet als Eingabe den obligatorischen Prozedurnamen und den optionalen Prozedurtyp. Dann wird eine Anfrage an den Datenbankserver gestellt, durch die festgestellt wird, ob die betroffene Prozedur in der Datenbank gespeichert ist. Wenn es der Fall ist, wird sie gelöscht.

Option A=4B=4

Zusammenfassung:

Eingabe: Prozedurnamen (obligatorisch), Prozedurtyp (optional)

Automatisierungsmodus:

`korpo -s "Datenbankname" A=4 B=4 "Prozedurnamen" "Prozedurtyp" -e`

oder

`korpo -s "Datenbankname" A=4 B=4 "Prozedurnamen" -e`

- **5: Haupttyp löschen**

Mit diesem Menüpunkt bekommt der Benutzer ein Werkzeug zur Hand, Datensätze aus der Tabelle HAUPTTYP zu eliminieren. Dazu ist die Routine `db_haupttyp_löschen` zuständig, die eine Haupttypkennzeichnung als Eingabe entgegennimmt, die eindeutig einen Haupttyp identifiziert. Als nächstes wird auf die Existenz des zu löschenden Haupttyps geprüft. Im positiven Fall wird er gelöscht.

Option A=4B=5

Zusammenfassung:

Eingabe: Haupttypkennzeichnung (obligatorisch)

Automatisierungsmodus:

korpo -s "Datenbankname" A=4 B=5 "Haupttypkennzeichnung" -e

- **6: Untertyp löschen**

Bei diesem Menüpunkt fungiert die Routine db_untertyp_löschen anstelle der Routine db_haupttyp_löschen. Als Eingabe wird eine Untertypkennzeichnung erwartet, die den zu löschenden Untertyp identifiziert.

Option A=4B=6

Zusammenfassung:

Eingabe: Untertypkennzeichnung (obligatorisch)

Automatisierungsmodus:

korpo -s "Datenbankname" A=4 B=6 "Untertypkennzeichnung" -e

- **7: Quelle löschen**

Mit Hilfe dieses Menüpunkts können Datensätze aus der Tabelle QUELLE gelöscht werden. Hierbei wird die Routine db_quelle_löschen aufgerufen. Zwei Eingaben sind zu tätigen: die 1. Bezeichnung (obligatorisch) und die 2. Bezeichnung des zu löschenden Quelle- Datensatzes. Es wird weiterhin überprüft, ob ein entsprechender Datensatz in der Tabelle QUELLE vorliegt. Dieser Datensatz wird gelöscht, sofern er gefunden wird.

Option A=4B=7

Zusammenfassung:

Eingabe: 1. Bezeichnung (obligatorisch), 2. Bezeichnung (optional)

Automatisierungsmodus:

korpo -s "Datenbankname" A=4 B=7 "Bezeichnung 1" "Bezeichnung 2" -e
oder

korpo -s "Datenbankname" A=4 B=7 "Bezeichnung 1" -e

- **8: Dateizuordnung löschen**

Dieser Menüpunkt dient dem Löschen von Datensätzen, die in denjenigen Tabellen gespeichert sind und in Beziehung mit Datensätzen in der Tabelle FILE stehen, wobei eine Sprachsignaldatei in der Tabelle FILE der Ausgangspunkt ist. Da es mehrere solche Tabellen gibt, wie z.B. FILE_FO_ZUORDNUNG, FILE_PRO_ZUORDNUNG usw., wird dieser Menüpunkt weiter aufgeteilt und führt deswegen zu einem tiefer angegliederten Menüsystem, das weiter unten besprochen wird.

Option A=4B=8

Menüsystem

- **9: Formatzuordnung löschen**

Dieser Menüpunkt wird ausgewählt zum Löschen von Daten in der Tabelle FILE_FO_ZUORDNUNG, sofern eine Formatkennzeichnung als Ausgangskriterium angegeben wird. D.h. als erstes muß eine Formatkennzeichnung unbedingt eingegeben werden, die als Eingabe der Routine `db_format_zuordnung_loeschen` dient. Danach wird geprüft, ob das entsprechende Format in der Tabelle FORMAT zu finden ist. Im positiven Fall wird der Benutzer aufgefordert, einen Dateinamen einzugeben. Diese Angabe ist jedoch optional. Es kommen zwei Fälle in Betracht:

- Ein Dateiname wird eingegeben:
In diesem Fall wird derjenige Datensatz in der Tabelle FILE_FO_ZUORDNUNG gelöscht, der den eingegebenen Dateinamen und die eingegebene Formatkennzeichnung hat.
- `< RETURN >` wird gedrückt (keine Angabe):
In diesem Fall kommt eine Rückfrage, ob alle Datensätze in der Tabelle FILE_FO_ZUORDNUNG gelöscht werden sollen, die nur die eingegebene Formatkennzeichnung beinhalten. Falls die Antwort Ja lautet, werden sie alle gelöscht. Daher soll man vorsichtig ein mit der Ja- Bestätigung.

Option A=4B=9

Zusammenfassung:

Eingabe: 1. Formatkennzeichnung (obligatorisch), Dateiname (optional))

Automatisierungsmodus:

korpo -s "Datenbankname" A=4 B=9 "Formatkennzeichnung" "Dateinamen" -e
oder
korpo -s "Datenbankname" A=4 B=9 "Formatkennzeichnung" -e

• **10: Prozedurzuordnung löschen**

Dieser Menüpunkt wird verwendet, wenn Datensätze aus der Tabelle FILE_PRO_ZUORDNUNG ausgehend von einem Prozedurnamen gelöscht werden sollen. Dazu wird die Routine `db_prozedur_zuordnung_loeschen` gestartet. Zuerst soll ein Prozedurname unbedingt eingegeben werden, die Angabe über einen Prozedurtyp ist im Gegensatz dazu nur optional. Anhand des eingegebenen Prozedurnamens (und des Prozedurtyps) kann bestimmt werden, ob die Prozedur in der Datenbank existiert. Falls sie gefunden wird, wird anschließend nach einem Dateinamen gefragt. Auch hier gilt es zu unterscheiden:

- Ein Dateiname wird eingegeben:
In diesem Fall wird derjenige Datensatz in der Tabelle FILE_PRO_ZUORDNUNG gelöscht, der den eingegebenen Dateinamen und den eingegebenen Prozedurnamen hat.
- `< RETURN >` wird gedrückt (keine Angabe):
In diesem Fall kommt eine Rückfrage, ob alle Datensätze in der Tabelle FILE_PRO_ZUORDNUNG gelöscht werden sollen, die nur den eingegebenen Prozedurnamen beinhalten. Falls die Antwort Ja lautet, werden sie gelöscht.

Option A=4B=10

Zusammenfassung:

Eingabe: Prozedurnamen (obligatorisch), Prozedurtyp (optional), Dateiname (optional)

Automatisierungsmodus:

korpo -s "Datenbankname" A=4 B=10 "Prozedurnamen" "Prozedurtyp" -e

oder

korpo -s "Datenbankname" A=4 B=10 "Prozedurnamen" -e

• 11: Prozedur–Dateiextension–Zuordnung löschen

Mit Hilfe dieses Menüpunkts kann man gewünschte Datensätze in der Tabelle FILEEXT_PROZEDUR_ZU löschen. Den Ausgangspunkt stellt die Eingabe einer Dateiextension dar, die von der verwendeten Routine `db_fileext_prozedur_zu_loeschen` erwartet wird. Eine Dateiextension muß aber auf jeden Fall eingegeben werden. Als nächstes kommt die Aufforderung, einen Prozedurnamen einzugeben - diese Angabe ist allerdings optional. Es werden auch hier zwei Fälle unterschieden:

- Angabe über Prozedurnamen fehlt:

In diesem Fall wird der Benutzer gefragt, ob alle Datensätze in der Tabelle FILEEXT_PROZEDUR_ZU gelöscht werden sollen, die nur die eingegebene Dateiextension enthalten. Wenn die Antwort Ja lautet, werden alle unwiederbringlich gelöscht.

- Angabe über Prozedurnamen vorhanden:

Nun werden lediglich diejenige Datensätze aus der Tabelle FILEEXT_PROZEDUR_ZU gelöscht, die den Angaben im Ganzen entsprechen.

Option A=4B=11

Zusammenfassung:

Eingabe: Dateiextension (obligatorisch), Prozedurnamen (optional)

Automatisierungsmodus:

korpo -s "Datenbankname" A=4 B=11 "Dateiextension" "Prozedurnamen" -e

oder

korpo -s "Datenbankname" A=4 B=11 "Dateiextension" -e

• 12: Prozedur–Dateiextension–Zuordnung Ziel löschen

Mit Hilfe dieses Menüpunkts kann man ebenfalls gewünschte Datensätze in der Tabelle FILEEXT_PROZEDUR_ZU löschen. Den Ausgangspunkt stellt in diesem Fall aber die Eingabe einer Zieldateiextension dar, die von der verwendeten Routine `db_fileext_prozedur_zu_ziel_loeschen` erwartet wird und dem Datenfeld "outext" in der Tabelle FILEEXT_PROZEDUR_ZU entspricht. Eine Zieldateiextension muß aber auf jeden Fall eingegeben werden. Als nächstes kommt die Aufforderung, einen Prozedurnamen einzugeben -

diese Angabe ist allerdings optional. Es werden auch hier zwei Fälle unterschieden:

- Angabe über Prozedurnamen fehlt:

In diesem Fall wird der Benutzer gefragt, ob alle Datensätze in der Tabelle FILEEXT_PROZEDUR_ZU gelöscht werden sollen, die nur die eingegebene Zieldateiextension enthalten. Wenn die Antwort Ja lautet, werden alle unwiederbringlich gelöscht.

- Angabe über Prozedurnamen vorhanden:

Nun werden lediglich diejenige Datensätze aus der Tabelle FILEEXT_PROZEDUR_ZU gelöscht, die den Angaben im Ganzen entsprechen.

Option A=4B=12

Zusammenfassung:

Eingabe: Zielformat (obligatorisch), Prozedurnamen (optional)

Automatisierungsmodus:

korpo -s "Datenbankname" A=4 B=12 "Zielformat" "Prozedurnamen" -e
oder

korpo -s "Datenbankname" A=4 B=12 "Zielformat" -e

• **13: Haupttypzuordnung löschen**

Dieser Menüpunkt ist zuständig für das Löschen von Datensätzen aus der Tabelle FILE_HT_ZUORDNUNG unter der Regie der Routine db_haupttyp_zuordnung_loeschen. Nun muß eine Haupttypkennzeichnung eingegeben werden. Anschließend wird ein Dateiname erwartet, der jedoch nur optional ist (Aufforderung kann man < RETURN > beantwortet werden). Es gibt wie üblich zwei Unterscheidungsfälle:

- Angabe über Dateinamen fehlt:

In diesem Fall wird der Benutzer gefragt, ob alle Datensätze in der Tabelle FILE_HT_ZUORDNUNG gelöscht werden sollen, welche die eingegebene Haupttypkennzeichnung eindeutig identifizieren. Wenn die Antwort Ja lautet, werden diese Datensätze gelöscht.

- Angabe über Dateinamen vorhanden:

Nun werden lediglich diejenige Datensätze aus der Tabelle FILE_HT_ZUORDNUNG gelöscht, die den Angaben im Ganzen entsprechen.

Option A=4B=13

Zusammenfassung:

Eingabe: Haupttypkennzeichnung (obligatorisch), Dateinamen (optional)

Automatisierungsmodus:

korpo -s "Datenbankname" A=4 B=13 "Haupttypkennzeichnung" "Dateinamen" -e
oder

korpo -s "Datenbankname" A=4 B=13 "Haupttypkennzeichnung" -e

• **14: Untertypzuordnung löschen**

Analog zum Menüpunkt zum Löschen von Daten in der Tabelle FILE_UT_ZUORDNUNG verläuft hier die hierbei aktivierte Routine db_untertyp_loeschen. Angabe über Untertypkennzeichnung ist unbedingt erforderlich, ein Dateiname ist aber optional. Je nach Angaben ausgewählte Datensätze aus der Tabelle FILE_UT_ZUORDNUNG gelöscht.

Option A=4B=14

Zusammenfassung:

Eingabe: Untertypkennzeichnung (obligatorisch), Dateinamen (optional)

Automatisierungsmodus:

korpo -s "Datenbankname" A=4 B=14 "Untertypkennzeichnung" "Dateinamen" -e
oder

korpo -s "Datenbankname" A=4 B=14 "Untertypkennzeichnung" -e

- **15: Quellezuordnung löschen**

Dieser Menüpunkt wird verwendet, um Datensätze aus der Tabelle FILE_QU_ZUORDNUNG zu löschen. Dazu dient die Routine db_quelle_zuordnung_loeschen. Der Benutzer wird aufgefordert, die erste und die zweite Bezeichnung einer Quelle einzugeben, wobei die erste obligatorisch und die zweite optional ist. Danach wird ein Dateiname erwartet, diese Angabe kann wie bisher mit < RETURN > quittiert werden. Fehlt diese Angabe, kommt eine Rückfrage, ob alle Datensätze in der Tabelle FILE_QU_ZUODNUNG, die nur der Angabe über Bezeichnungen genügen, gelöscht werden sollen. Im positiven Fall werden diese gelöscht. Ist ein Dateiname vorhanden, werden nur Datensätze gelöscht, die sowohl die Bezeichnungen der Quelle als auch den Dateinamen beinhalten.

Option A=4B=15

Zusammenfassung:

Eingabe: 1. Bezeichnung (obligatorisch), 2. Bezeichnung (optional), Dateiname (optional)

Automatisierungsmodus:

korpo -s "Datenbankname" A=4 B=15 "Bezeichnung 1" "Bezeichnung 2"
"Dateinamen" -e

oder

korpo -s "Datenbankname" A=4 B=15 "Bezeichnung 1" "Dateinamen" -e

oder

korpo -s "Datenbankname" A=4 B=15 "Bezeichnung 1" -e

- **16: Zugriffszuordnung löschen**

Mit diesem Menüpunkt kann die Routine db_zugriff_zuordnung_loeschen gestartet werden, deren Aufgabe darin besteht, Datensätze aus der Tabelle FILE_ZUG_ZUORDNUNG zu löschen. Der Benutzer muß eine Zugriffs-ID eingeben, die ja ein bestimmtes Zugriffsrecht eindeutig kennzeichnet. Anschließend wird ein Dateiname erwartet, der jedoch nur optional ist. Nun wird überprüft, ob die eingegebene Zugriffs-ID in der Datenbank vorhanden ist. Soll es der Fall sein, wird wie folgt vorgegangen:

- Ein Dateiname ist nicht vorhanden:

Es kommt noch mal die Rückfrage, ob all diejenige Datensätze aus der Tabelle FILE_ZUG_ZUORDNUNG gelöscht werden sollen, die die eingegebene Zugriffs-ID haben. Bei Ja-Antwort werden die qualifizierten gelöscht.

- Ein Dateiname ist vorhanden:

Nur Datensätze, die den Angaben genügen, werden gelöscht.

Option A=4B=16

Zusammenfassung:

Eingabe: Zugriffs-ID (obligatorisch), Dateiname (optional)

Automatisierungsmodus:

```
korpo -s "Datenbankname" A=4 B=16 "Zugriffs-ID" "Dateinamen" -e
```

oder

```
korpo -s "Datenbankname" A=4 B=16 "Zugriffs-ID" -e
```

- **17: Tabelle komplett löschen**

Anhand dieses Menüpunkts kann der Inhalt einer Tabelle komplett gelöscht werden. Dabei wird die Routine `delete_tabelle_allgemein()` verwendet, die einen Tabellennamen als übergebenes Argument bekommt. Der Tabellenname ist auf jeden Fall einzugeben.

Option A=4B=17

Zusammenfassung Eingabe: Tabellenname (obligatorisch)

Automatisierungsmodus:

```
korpo -s "Datenbankname" A=4 B=17 "Tabellenname" -e
```

3.4.6 Menü A5_B

- **1: Änderungen-Datei**

Dieser Menüpunkt ermöglicht Änderungsoperationen auf Datenbestände der Tabelle FILE. Es gibt mehrere Positionen, an denen Änderungen durchgeführt werden können. Aus diesem Grund besteht dieser Menüpunkt aus weiteren Untermenüpunkten, über die an nächstfolgenden Stellen erläutert wird.

Option A=5B=1

Menüsystem

- **2: Änderungen-Format**

Mit diesem Menüpunkt kann die Formatkennzeichnung eines Eintrags in der Tabelle FORMAT modifiziert werden, indem die Routine `db_change_format` aufgerufen wird. Die zu modifizierende Formatkennzeichnung und die neue Formatkennzeichnung müssen natürlich unbedingt angegeben werden. Danach wird überprüft, ob die zu modifizierende Formatkennzeichnung tatsächlich in der Tabelle FORMAT vorhanden ist. Soll es der Fall sein, findet die Änderung statt.

Option A=5B=2

Zusammenfassung:

Eingabe: zu ändernde Formatkennzeichnung (obligatorisch)

neue Formatkennzeichnung (obligatorisch)

Automatisierungsmodus:

```
korpo -s "Datenbankname" A=5 B=2 "alte Formatkennzeichnung"
```

```
"neue Formatkennzeichnung" -e
```

- **3: Änderungen-Prozedur**

Genau wie beim Menüpunkt zur Änderung von Datensätzen in der Tabelle FILE führt dieser Menüpunkt zu weiteren Menüpunkten, die der Modifizierung von Daten in der Tabelle PROZEDUR dienen.

Option A=5B=3

Menüsystem

- **4: Änderungen–Haupttyp**

Zur Änderung von Daten in der Tabelle HAUPTTYP wird dieser Menüpunkt angewählt, indem die Routine db_change_haupttyp aktiviert wird. Auch hier sind zwei Angaben unbedingt erforderlich: die zu modifizierende Haupttypkennzeichnung und die neue Haupttypkennzeichnung. Es wird weiterhin auf das Vorhandensein des Datensatzes mit der zu modifizierenden Kennzeichnung in der Tabelle HAUPTTYP geprüft. Im Erfolgsfall wird die Änderung vorgenommen.

Option A=5B=4

Zusammenfassung:

Eingabe: zu ändernde Haupttypkennzeichnung (obligatorisch)

neue Haupttypkennzeichnung (obligatorisch)

Automatisierungsmodus:

korpo -s "Datenbankname" A=5 B=4 "alte Haupttypkennzeichnung"

"neue Haupttypkennzeichnung" -e

- **5: Änderungen–Untertyp**

Bei diesem Menüpunkt wird ähnlich vorgegangen wie bei der Änderung von Dateneinträgen in der Tabelle HAUPTTYP. In diesem Fall kommt die Routine db_change_untertyp ins Spiel. Eingabewerte sind die zu modifizierende Untertypkennzeichnung und die neue Untertypkennzeichnung. Beide Werte sind obligatorisch. Es wird ebenfalls vor der Änderung geprüft, ob der zu ändernde Datensatz in der Tabelle UNTERTYP vorliegt.

Option A=5B=5

Zusammenfassung:

Eingabe: zu ändernde Untertypkennzeichnung (obligatorisch)

neue Untertypkennzeichnung (obligatorisch)

Automatisierungsmodus:

korpo -s "Datenbankname" A=5 B=4 "alte Untertypkennzeichnung"

"neue Untertypkennzeichnung" -e

- **6: Änderungen–Quelle**

Hierbei werden qualifizierte Datensätze aus der Tabelle QUELLE mit Hilfe der Routine db_change_quelle geändert. Die erste Bezeichnung des zu ändernden Quelle-Datensatzes muß unbedingt eingegeben werden. Überdies ist die neue erste Bezeichnung obligatorisch, die neue zweite Bezeichnung ist optional. Je nach Angabe wird zunächst überprüft, ob der zu ändernde Datensatz in der Datenbank vorhanden ist oder nicht. Im positiven Fall wird die Änderung den Angaben entsprechend

vorgenommen.

Option A=5B=6

Zusammenfassung:

Eingabe: zu ändernde 1. Bezeichnung (obligatorisch)

neue 1. Bezeichnung (obligatorisch), neue 2. Bezeichnung (optional)

Automatisierungsmodus:

korpo -s "Datenbankname" A=5 B=6 "alte Bezeichnung" "neue Bezeichnung 1"

"neue Bezeichnung 2" -e

korpo -s "Datenbankname" A=5 B=6 "alte Bezeichnung" "neue Bezeichnung 1" -e

• **7: Änderungen-Datei-Zuordnung**

Es gibt nämlich mehrere Tabellen, bei denen Beziehungen zwischen Sprachsignaldateien (in der Tabelle FILE) und anderen Tabellen ausgedrückt werden (z.B. FILE_FO_ZUORDNUNG, FILE_PRO_ZUORDNUNG ...). Aus diesem Grund führt dieser Menüpunkt zu weiteren Menüpunkten, bei denen die Änderungen an den entsprechenden Tabellen vorgenommen werden können.

Option A=5B=7 Menüsystem

• **8: Änderungen-Format-Zuordnung**

Hierbei können Änderungen an Datenbeständen in der Tabelle FILE_FO_ZUORDNUNG durchgeführt werden, indem die Zuordnung eines bestimmten Formats zu Sprachsignaldateien geändert werden soll. Die verwendete Routine in diesem Fall ist `db_change_format_zuordnung`. Als erstes wird der Benutzer aufgefordert, eine Formatkennzeichnung einzugeben. Dann wird überprüft, ob das angegebene Format in der Datenbank vorliegt. Ist alles in Ordnung, werden anschließend zwei Angaben benötigt: den zu modifizierenden Dateinamen und den neuen Dateinamen. Der neue Dateiname ist dabei unbedingt anzugeben, der zu modifizierende Dateiname ist jedoch optional. Nun findet die Prüfung statt, ob der neue Dateiname in der Datenbank vorkommt. Soll es nicht der Fall sein, wird die Routine mit einer Fehlermeldung abgebrochen. Zwei Fälle sind zu unterscheiden:

- der zu modifizierende Dateiname ist nicht eingegeben:

In diesem Fall wird eine Rückfrage an den Benutzer gestellt, ob diejenige Datensätze in der Tabelle FILE_FO_ZUORDNUNG geändert werden sollen, bei denen die eingegebene Formatkennzeichnung partizipiert ist. Bei Ja-Antwort wird an diesen Datensätzen der alte Dateiname durch den neuen angegebenen Dateinamen ersetzt. Auf diese Weise wird auf einmal die Zuordnung von mehreren Sprachsignaldateien zu einem bestimmten Format komplett dadurch geändert, indem die darin enthaltenen Sprachsignaldateien durch die neue ersetzt werden.

- der zu modifizierende Dateiname ist angegeben:

Nun werden nur diejenige Datensätze modifiziert, an denen sowohl die Formatkennzeichnung als auch der zu modifizierende Dateiname eine Beziehung eingeht. Somit wird die Zuordnung eines bestimmten Formats zu einer bestimmten Sprachsignaldatei geändert, indem diese Zuordnung fortan auf der neuen Sprachsignaldatei basiert.

Option A=5B=8

Zusammenfassung:

Eingabe: Formatkennzeichnung (obligatorisch)
 zu modifizierenden Dateinamen (optional)
 neuen Dateinamen (obligatorisch)

Automatisierungsmodus:

korpo -s "Datenbankname" A=5 B=8 "Formatkennzeichnung" "Dateinamen"
 "neuen Dateinamen" -e

korpo -s "Datenbankname" A=5 B=8 "Formatkennzeichnung" "neuen Dateinamen" -e

- **9: Änderungen-Prozedur-Zuordnung**

Zur Änderung von Datenbeständen in der Tabelle FILE_PRO_ZUORDNUNG wird dieser Menüpunkt ausgewählt. Dafür wird die Routine db_change_prozedur_zuordnung herangezogen. Als Ausgangspunkt dient ein Prozedurname, der unbedingt anzugeben ist. Es wird als nächstes geprüft, ob diese Prozedur in der Tat in der Datenbank vorkommt. Soll es der Fall sein, sind Angaben über den zu modifizierenden Dateinamen und den neuen Dateinamen an der Reihe, wobei die erste Angabe nur optional ist. Danach wird ebenfalls überprüft, ob der neue Dateiname in der Datenbank vorhanden ist. Im positiven Fall gibt es zwei Unterscheidungsfälle:

- der zu modifizierende Dateiname ist nicht angegeben:
 In diesem Fall wird eine Rückfrage an den Benutzer gestellt, ob diejenige Datensätze in der Tabelle FILE_PRO_ZUORDNUNG geändert werden sollen, bei denen der eingegebene Prozedurname partizipiert ist. Bei Ja-Antwort wird an diesen Datensätzen der alte Dateiname durch den neuen angegebenen Dateinamen ersetzt. Auf diese Weise wird auf einmal die Zuordnung von mehreren Sprachsignaldateien zu einer bestimmten Prozedur komplett dadurch geändert, indem die darin enthaltenen Sprachsignaldateien durch die neue ersetzt werden.
- der zu modifizierende Dateiname ist angegeben:
 Nun werden nur diejenige Datensätze modifiziert, an denen sowohl der Prozedurname als auch der zu modifizierende Dateiname eine Beziehung eingeht. Somit wird die Zuordnung einer bestimmten Prozedur zu einer bestimmten Sprachsignaldatei geändert, indem diese Zuordnung fortan auf der neuen Sprachsignaldatei basiert.

Option A=5B=9

Zusammenfassung:

Eingabe: Prozedurnamen (obligatorisch)
 zu modifizierenden Dateinamen (optional)
 neuen Dateinamen (obligatorisch)

Automatisierungsmodus:

korpo -s "Datenbankname" A=5 B=9 "Prozedurnamen" "Dateinamen"
 "neuen Dateinamen" -e

korpo -s "Datenbankname" A=5 B=9 "Prozedurnamen" "neuen Dateinamen" -e

- **10: Änderungen-Haupttyp-Zuordnung**

Mit diesem Menüpunkt kann man ausgewählte Datensätze in der Tabelle FILE_HT_ZUORDNUNG unter Mitwirkung der Routine db_change_haupttyp_zuordnung ändern. Die Auswahl erfolgt durch Angaben

über Haupttypkennzeichnung und Dateinamen. Dementsprechend muß unbedingt eine Haupttypkennzeichnung zuerst angegeben werden. Anschließend wird überprüft, ob der entsprechende Haupttyp in der Datenbank zu finden ist. Soll es der Fall sein, kommt als nächstes die Aufforderung, den zu ändernden und den neuen Dateinamen anzugeben, wobei der erstgenannte Dateiname nur optional ist. Weiterhin wird in der Datenbank n

ach dem neuen Dateinamen nachgeschaut, ob er in der Datenbank vorkommt. Im positiven kommen zwei Unterscheidungsfälle in Betracht:

- der zu modifizierende Dateiname ist nicht angegeben:

In diesem Fall wird eine Rückfrage an den Benutzer gestellt, ob diejenige Datensätze in der Tabelle FILE_HT_ZUORDNUNG geändert werden sollen, bei denen die eingegebene Haupttypkennzeichnung partizipiert ist. Bei Ja-Antwort wird an diesen Datensätzen der alte Dateiname durch den neuen angegebenen Dateinamen ersetzt. Auf diese Weise wird auf einmal die Zuordnung von mehreren Sprachsignaldateien zu einem bestimmten Haupttyp komplett dadurch geändert, indem die darin enthaltenen Sprachsignaldateien durch die neue ersetzt werden.

- der zu modifizierende Dateiname ist angegeben:

Nun werden nur diejenige Datensätze modifiziert, an denen sowohl die Haupttypkennzeichnung als auch der zu modifizierende Dateiname eine Beziehung eingeht. Somit wird die Zuordnung eines bestimmten Haupttyps zu einer bestimmten Sprachsignaldatei geändert, indem diese Zuordnung fortan auf der neuen Sprachsignaldatei basiert.

Option A=5B=10

Zusammenfassung:

Eingabe: Haupttypkennzeichnung (obligatorisch)
 zu modifizierenden Dateinamen (optional)
 neuen Dateinamen (obligatorisch)

Automatisierungsmodus:

```
korpo -s "Datenbankname" A=5 B=10 "Haupttypkennzeichnung" "Dateinamen"  

"neuen Dateinamen" -e
```

```
korpo -s "Datenbankname" A=5 B=10 "Haupttypkennzeichnung" "neuen Dateinamen"  

-e
```

• 11: Änderungen–Untertyp–Zuordnung

Bei dieser Auswahl zeigt das Programm einen ähnlichen Ablauf wie bei der Änderung von Datensätzen in der Tabelle FILE_HT_ZUORDNUNG wie oben beschrieben. In diesem Fall wird die Routine db_change_untertyp_zuordnung zu Hilfe gezogen. Eine Untertypkennzeichnung muß eingegeben werden. Es gilt die gleiche Bedingung für Angaben über Dateinamen. Es wird auch festgestellt, ob der Untertyp und der neue Dateiname in der Datenbank vorkommen. Wenn dies zutrifft, wird wie folgt vorgegangen:

- der zu modifizierende Dateiname ist nicht angegeben:

In diesem Fall wird eine Rückfrage an den Benutzer gestellt, ob diejenige Datensätze in der Tabelle FILE_UT_ZUORDNUNG geändert werden sollen, bei denen die eingegebene Untertypkennzeichnung partizipiert ist. Bei Ja-Antwort wird an diesen Datensätzen der alte Dateiname durch den neuen angegebenen Dateinamen ersetzt. Auf diese Weise wird auf einmal die Zuordnung von mehreren Sprachsignaldateien zu einem bestimmten Untertyp komplett dadurch geändert, indem die darin enthaltenen Sprachsignaldateien durch die neue ersetzt werden.

- der zu modifizierende Dateiname ist angegeben:
Nun werden nur diejenige Datensätze modifiziert, an denen sowohl die Untertypkennzeichnung als auch der zu modifizierende Dateiname eine Beziehung eingeht. Somit wird die Zuordnung eines bestimmten Untertyps zu einer bestimmten Sprachsignaldatei geändert, indem diese Zuordnung fortan auf der neuen Sprachsignaldatei basiert.

Option A=5B=11

Zusammenfassung:

Eingabe: Untertypkennzeichnung (obligatorisch)
zu modifizierenden Dateinamen (optional)
neuen Dateinamen (obligatorisch)

Automatisierungsmodus:

korpo -s "Datenbankname" A=5 B=11 "Untertypkennzeichnung" "Dateinamen"
"neuen Dateinamen" -e

korpo -s "Datenbankname" A=5 B=11 "Untertypkennzeichnung" "neuen Dateinamen" -e

• **12: Änderungen–Quelle–Zuordnung**

Mit Hilfe dieses Menüpunkts können Änderungen in der Tabelle FILE_QU_ZUORDNUNG vorgenommen werden. Dabei wird die Routine db_change_quelle_zuordnung verwendet, die zunächst Angaben über Quellenbezeichnungen erwartet, wobei die 1. Bezeichnung obligatorisch ist. Unmittelbar danach wird festgestellt, ob eine Quelle in der Datenbank der Angabe entspricht. Wenn dies zutrifft, werden Angaben über Dateinamen gemacht. Es sind wie üblich zwei Dateinamen anzugeben: der zu modifizierende und der neue Dateiname. Der neue Dateiname ist aber auf jeden Fall einzugeben. Nun findet die Prüfung auf das Vorhandensein der neuen Sprachsignaldatei in der Datenbank statt. Soll es der Fall sein, sind zwei Fälle zu unterscheiden:

- der zu modifizierende Dateiname ist nicht angegeben:
In diesem Fall wird eine Rückfrage an den Benutzer gestellt, ob diejenige Datensätze in der Tabelle FILE_QU_ZUORDNUNG geändert werden sollen, bei denen die eingegebenen Quellenbezeichnungen partizipiert sind. Bei Ja–Antwort wird an diesen Datensätzen der alte Dateiname durch den neuen angegebenen Dateinamen ersetzt. Auf diese Weise wird auf einmal die Zuordnung von mehreren Sprachsignaldateien zu einer bestimmten Quelle komplett dadurch geändert, indem die darin enthaltenen Sprachsignaldateien durch die neue ersetzt werden.
- der zu modifizierende Dateiname ist angegeben:
Nun werden nur diejenige Datensätze modifiziert, an denen sowohl die Quellenbezeichnungen als auch der zu modifizierende Dateiname eine Beziehung eingeht. Somit wird die Zuordnung einer bestimmten Quelle zu einer bestimmten Sprachsignaldatei geändert, indem diese Zuordnung fortan auf der neuen Sprachsignaldatei basiert.

Option A=5B=12

Zusammenfassung:

Eingabe: 1. Quellenbezeichnung (obligatorisch)
2. Quellenbezeichnung (optional)
zu modifizierenden Dateinamen (optional)
neuen Dateinamen (obligatorisch)

Automatisierungsmodus:

korpo -s "Datenbankname" A=5 B=12 "Bezeichnung 1" "Bezeichnung 2"

"Dateinamen" "neuen Dateinamen" -e

korpo -s "Datenbankname" A=5 B=12 "Bezeichnung 1" "Dateinamen"

"neuen Dateinamen" -e

korpo -s "Datenbankname" A=5 B=12 "Bezeichnung 1" "neuen Dateinamen" -e

• 13: Änderungen-Prozedur-Dateiextension-Zuordnung

Die Tabelle FILEEXT_PROZEDUR_ZU stellt die Beziehung zwischen einer Dateiextension und einer Sprachsignaldatei dar. Somit kann bestimmt werden, welche Prozeduren einer Dateiextension zugeordnet sind. Diese Information ist erforderlich bei der Formatumwandlung von Sprachsignaldateien. Änderungen können an zwei Stellen vorgenommen werden, nämlich, an den partizipierenden Dateiextensionen (Quelldateiextension und Zieldateiextension) und dem partizipierenden Prozedurnamen. Dies wird Rechnung getragen, indem dieser Menüpunkt zu zwei Menüpunkten führt, die später beschrieben werden.

Option A=5B=13

Menüsystem

• 14: Änderungen-Zugriff-Zuordnung

Dieser Menüpunkt wird verwendet bei Änderungen in der Tabelle FILE_ZUG_ZUORDNUNG. Dabei wird die Routine db_change_zugriff_zuordnung aktiviert, die u.a. eine Zugriffs-ID erwartet. Nach der Eingabe wird festgestellt, ob die angegebene Zugriffs-ID gültig ist. Danach sind Angaben über Dateinamen zu tätigen, die natürlich auf ihre Gültigkeit geprüft werden. Eine Angabe ist gültig, wenn die entsprechenden Daten in der Datenbank vorkommen. Soll alles in Ordnung sein, wird die gewünschte Änderung vorgenommen. Dies geschieht auf zwei Wege:

- der zu modifizierende Dateiname ist nicht angegeben:

In diesem Fall wird eine Rückfrage an den Benutzer gestellt, ob diejenige Datensätze in der Tabelle FILE_ZUG_ZUORDNUNG geändert werden sollen, bei denen die eingegebene Zugriffs-ID partizipiert ist. Bei Ja-Antwort wird an diesen Datensätzen der alte Dateiname durch den neuen angegebenen Dateinamen ersetzt. Auf diese Weise wird auf einmal die Zuordnung von mehreren Sprachsignaldateien zum neuen Zugriffsrecht komplett dadurch geändert, indem die darin enthaltenen Sprachsignaldateien durch die neue ersetzt werden.

- der zu modifizierende Dateiname ist angegeben:

Nun werden nur diejenige Datensätze modifiziert, an denen sowohl die eingegebene Zugriffs-ID als auch der zu modifizierende Dateiname eine Beziehung eingeht. Somit wird die Zuordnung eines bestimmten Zugriffsrechts zu einer bestimmten Sprachsignaldatei geändert, indem diese Zuordnung fortan auf der neuen Sprachsignaldatei basiert.

Option A=5B=14

Zusammenfassung:

Eingabe: Zugriffs-ID (obligatorisch)

zu modifizierenden Dateinamen (optional)

neuen Dateinamen (obligatorisch)

Automatisierungsmodus:

korpo -s "Datenbankname" A=5 B=14 "Zugriffs-ID" "Dateinamen"

"neuen Dateinamen" -e
 korpo -s "Datenbankname" A=5 B=14 "Zugriffs-ID" "neuen Dateinamen" -e

3.4.7 Menü A1B1_C

Grundsätzlich handelt es sich bei diesem Menüsystem um die Formulierung von Abfragen, die durch ein SQL-Statement ausgedrückt und an den Datenbankserver übergeben werden.

- **1: Alle Dateien**

Mit diesem Menüpunkt können alle Sprachsignaldateien in der Tabelle FILE zusammen mit der zugeordneten Prozedur und der assoziierten Quelle aufgelistet werden. Dabei kommt die Routine `ab_file_alle` zur Verwendung. Keine Eingabe ist benötigt. Wenn die Liste so lang ist, erfolgt die Rückfrage, ob man die Liste weiter anschauen möchte (nur im partiellen Automatisierungsmodus). Dabei wird nach dem Dateinamen für die Outputdatei gefragt, die ja das Ergebnis der Anfrage festhält und als Inputdatei für die später beschriebene Perl-Schnittstelle dient. Falls keine Angabe darüber erfolgt, wird die voreingestellte Datei "erg.dat" als Outputdatei fungieren. Im vollständigen Automatisierungsmodus wird ebenfalls die Outputdatei "erg.dat" herangezogen.

Option A=1B=1C=1

Zusammenfassung:

Eingabe: Outputdatei (optional)

Automatisierungsmodus:

korpo -s "Datenbankname" A=1 B=1 C=1 -e

- **2: Datei-Prozedur-Zuordnung**

Hierbei werden alle Datensätze in der Tabelle FILE_PRO_ZUORDNUNG mit Hilfe der Routine `ab_file_zugeordnet` aufgelistet. Keine Eingabe ist dazu notwendig. Die Auflistung enthält außer den Dateinamen auch die zugeordneten Prozedurnamen und die entsprechenden Quellen der Dateien.

Option A=1B=1C=2

Zusammenfassung:

Eingabe: keine

Automatisierungsmodus:

korpo -s "Datenbankname" A=1 B=1 C=2 -e

- **3: Datei-Haupttyp-Zuordnung**

Bei dieser Auswahl werden alle Datensätze in der Tabelle FILE_HT_ZUORDNUNG aufgelistet. Jeder dargestellte Eintrag in der Liste enthält außer dem Haupttyp auch die zugeordnete Sprachsignaldatei. Dabei wird die Routine `ab_file_haupttyp_zugeordnet` gestartet. Es wird auch keine Angabe erwartet.

Option A=1B=1C=3

Zusammenfassung:

Eingabe: keine

Automatisierungsmodus:

korpo -s "Datenbankname" A=1 B=1 C=3 -e

- **4: Datei-Untertyp-Zuordnung**

Bei dieser Auswahl werden alle Datensätze in der Tabelle FILE_UT_ZUORDNUNG aufgelistet. Jeder dargestellte Eintrag in der Liste enthält außer dem Untertyp auch die zugeordnete Sprachsignaldatei. Dabei wird die Routine `ab_file_untertyp_zugeordnet` gestartet.

Option A=1B=1C=4

Zusammenfassung:

Eingabe: keine

Automatisierungsmodus:

korpo -s "Datenbankname" A=1 B=1 C=4 -e

- **5: Datei-Format-Zuordnung**

Bei dieser Auswahl werden alle Datensätze in der Tabelle FILE_FO_ZUORDNUNG aufgelistet. Jeder dargestellte Eintrag in der Liste enthält außer der Formatkennzeichnung auch die zugeordnete Sprachsignaldatei. Dabei wird die Routine `ab_file_format_zugeordnet` gestartet.

Option A=1B=1C=5

Zusammenfassung:

Eingabe: keine

Automatisierungsmodus:

korpo -s "Datenbankname" A=1 B=1 C=5 -e

- **6: Dateiauswahl nach Namen**

Dieser Menüpunkt bedient sich der Routine `ab_file_name_auswahl` und liefert Informationen über die Sprachsignaldatei, deren Name unbedingt eingegeben werden muß. Auf diese Weise kann überprüft werden, ob eine bestimmte Sprachsignaldatei in der Datenbank vorkommt. Hier wird auch nach dem optionalen Namen der Outputdatei gefragt. Der voreingestellte Name der Outputdatei heißt "erg.dat".

Option A=1B=1C=6

Zusammenfassung:

Eingabe: Name der Sprachsignaldatei (obligatorisch)

Name der Outputdatei (optional)

Automatisierungsmodus:

korpo -s "Datenbankname" A=1 B=1 C=6 "Dateinamen" -e

- **7: Dateiauswahl nach Stamm**

Bei dieser Auswahl werden mit Hilfe der Routine `ab_file_pre_auswahl` anhand der eingegebenen "PRE-Daten" weitere Informationen über Sprachsignaldateien gewonnen. Es scheint hier angebracht zu sein, in Erinnerung zu rufen, daß die Tabelle FILE eine Spalte namens PRE hat, die vereinfacht gesagt den Stamm der Datei enthält (z.B. "dlf"). Dadurch hat man die Möglichkeit zu überprüfen, ob Sprachsignaldateien in der Datenbank existieren, die zu einem bestimmten Stamm gehören. Auch hier gilt es, einen optionalen Namen für die Outputdatei anzugeben. Falls keine Angabe erfolgt, wird die Datei namens "erg.dat" herangezogen.

Option A=1B=1C=7

Zusammenfassung:

Eingabe: PRE-Daten (obligatorisch) Name der Outputdatei (optional)

Automatisierungsmodus:

korpo -s "Datenbankname" A=1 B=1 C=7 "Stamm" -e

- **8: Dateiauswahl nach Datum**

Mit diesem Menüpunkt kann man bei Angabe eines Datums diejenige Sprachsignaldateien in der Tabelle FILE herausfinden, die solch ein Datum besitzen. Die dabei aktivierte Routine heißt `ab_file_datum_auswahl`, die ein obligatorisches Datum als Eingabe entgegennimmt.

Option A=1B=1C=8

Zusammenfassung:

Eingabe: Datum (obligatorisch)

Name der Outputdatei (optional)

Automatisierungsmodus:

korpo -s "Datenbankname" A=1 B=1 C=8 "Datum" -e

- **9: Dateiauswahl nach Typ (Dateiextension)**

Mit diesem Menüpunkt kann man bei Angabe einer Dateiextension diejenige Sprachsignaldateien in der Tabelle FILE herausfinden, die solch eine Extension besitzen. Die dabei aktivierte Routine heißt `ab_file_endung_auswahl`, die eine obligatorische Extension als Eingabe entgegennimmt. berdies wird nach dem Namen der Outputdatei gefragt, dessen Angabe jedoch optional ist. Falls keine Angabe erfolgt, heißt die verwendete Outputdatei "erg.dat".

Option A=1B=1C=9

Zusammenfassung:

Eingabe: Dateiextension (obligatorisch) Name der Outputdatei (optional)

Automatisierungsmodus:

korpo -s "Datenbankname" A=1 B=1 C=9 "Dateiextension" -e

- **10: Dateiauswahl nach Quelle**

Mit diesem Menüpunkt kann man bei Angabe einer Quelle diejenige Sprachsignaldateien in der Tabelle FILE herausfinden, die aus der angegebenen Quelle stammen. Die dabei aktivierte Routine heißt `ab_file_quelle_auswahl`, die eine obligatorische Quelle als Eingabe entgegennimmt. Überdies wird nach dem Namen der Outputdatei gefragt, dessen Angabe jedoch optional ist. Falls keine Angabe erfolgt, heißt die verwendete Outputdatei "erg.dat".

Option A=1B=1C=10

Zusammenfassung:

Eingabe: Quelle (obligatorisch) Name der Outputdatei (optional)

Automatisierungsmodus:

korpo -s "Datenbankname" A=1 B=1 C=10 "Quelle" -e

- **11: Dateiauswahl nach Pfad**

Mit diesem Menüpunkt kann man bei Angabe eines Pfades diejenige Sprachsignaldateien in der Tabelle FILE herausfinden, die sich unter dem angegebenen Pfad befinden, deren Datenfeld Pfad dem angegebenen Pfad entspricht. Die dabei aktivierte Routine heißt `ab_file_pfad_auswahl`, die einen obligatorischen Pfad als Eingabe entgegennimmt. Überdies wird nach dem Namen der Outputdatei gefragt, dessen Angabe jedoch optional ist. Falls keine Angabe erfolgt, heißt die verwendete Outputdatei "erg.dat".

Option A=1B=1C=11

Zusammenfassung:

Eingabe: Pfad (obligatorisch) Name der Outputdatei (optional)

Automatisierungsmodus:

korpo -s "Datenbankname" A=1 B=1 C=11 "Pfad" -e

- **12: Dateianzahl**

Dieser Menüpunkt ist hilfreich bei der Aufzählung der Datensätze in der Tabelle FILE. Dabei gibt es mehrere Möglichkeiten, Auswahlkriterien anzugeben. Infolgedessen wird dieser Menüpunkt weiter aufgeteilt in mehrere Menüpunkt, die später behandelt werden.

Option A=1B=1C=12

Menüsystem

- **13: Alle umgewandelten Dateien**

Dieser Menüpunkt listet alle Dateneinträge aus den Tabellen auf, die bei der Formatumwandlung die Namen der gerade erzeugten Ergebnisdateien im umzuwandelnden Format beinhalten. Dies sind die Tabellen FILE_SPH, FILE_PHONES,

FILE_WORDS und FILE_TEXT. Dabei wird die Routine `ab_file_output()` aktiviert, die den Namen einer der oben genannten Tabellen als Argument erwartet und eine Abfrage über den Inhalt der angegebenen Tabelle erstellt. Diese Angabe ist obligatorisch. Überdies wird der Benutzer auch aufgefordert, den Namen der Ergebnisdatei einzugeben. Allerdings ist hier die Angabe diesbezüglich optional. Die voreingestellte Ergebnisdatei ist "ergout.dat". D.h., das Ergebnis der Abfrage wird automatisch in die Datei `ergout.dat` geschrieben, sofern keine Angabe diesbezüglich getätigt wird.

Option A=1B=1C=13

Zusammenfassung:

Eingabe: Name der Outputtabelle (obligatorisch) Name der Outputdatei (optional)

Automatisierungsmodus:

korpo -s "Datenbankname" A=1 B=1 C=13 "Namen der Outputtabelle" -e

3.4.8 Menü A1B2_C

- **1: Alle Prozeduren**

Bei dieser Auswahl können alle vorhandenen Datensätze in der Tabelle PROZEDUR mit Hilfe der Routine `ab_prozedur_alle` aufgelistet werden.

Option A=1B=2C=1

Zusammenfassung:

Eingabe: Keine

Automatisierungsmodus:

korpo -s "Datenbankname" A=1 B=2 C=1 -e

- **2: Prozedur-Datei-Zuordnung**

Im Vergleich zum Menüpunkt "Datei-Prozedur-Zuordnung" werden auch hierbei Datensätze aus der Tabelle FILE_PRO_ZUORDNUNG aufgelistet. Allerdings stellt in diesem Fall die Tabelle PROZEDUR den Ausgangspunkt dar. D.h., es wird ausgegangen von der Tabelle PROZEDUR, indem für alle darin enthaltenen Datensätze untersucht wird, ob sie evtl. einer Sprachsignaldatei zugeordnet sind. Falls ja wird der entsprechende Datensatz in der Tabelle FILE_PRO_ZUORDNUNG angezeigt.

Option A=1B=2C=2

Zusammenfassung:

Eingabe: Keine

Automatisierungsmodus:

korpo -s "Datenbankname" A=1 B=2 C=2 -e

- **3: Prozedur-Dateiextension-Zuordnung**

Mit diesem Menüpunkt kann die Zuordnung zwischen Dateiextensionen und Prozedurnamen unter Mitwirkung der Routine `ab_fileext_prozedur_zu` ermittelt werden. Als Eingabe werden eine Dateiextension und ein Prozedurname erwartet. Beide Angaben müssen nicht gemacht werden. Allerdings muß eine Angabe getätigt werden (entweder über Dateiextension oder über Prozedurnamen). Nach richtiger Angabe gibt es folgende Unterscheidungsfälle:

- Angabe sowohl über Dateiextension als auch über Prozedurnamen:
In diesem Fall kann überprüft werden, ob die Zuordnung zwischen der Dateiextension und der Prozedur in der Datenbank vorhanden ist.
- Nur Angabe über Dateiextension:
In diesem Fall kann bestimmt werden, zu welchen Prozeduren die angegebene Dateiextension eine Beziehung eingeht. Mit anderem Wort, es wird ermittelt, welche Prozeduren auf die Sprachsignaldatei mit dieser Dateiextension angewandt werden können.
- Nur Angabe über Prozedurnamen:
In diesem Fall wird ermittelt, auf welche Dateiextension eine Prozedur angewandt werden kann.

Option A=1B=2C=3

Zusammenfassung:

Eingabe: Dateiextension und Prozedurnamen
oder
Dateiextension
oder
Prozedurnamen

Automatisierungsmodus:

korpo -s "Datenbankname" A=1 B=2 C=3 "Dateiextension"
"Prozedurnamen" -e

oder

korpo -s "Datenbankname" A=1 B=2 C=3 "Dateiextension" -e

oder

korpo -s "Datenbankname" A=1 B=2 C=3 "Prozedurnamen" -e

• **4: Prozedur-Dateiextension-Zuordnung-Ziel**

Mit diesem Menüpunkt kann die Zuordnung zwischen Zieldateiextensionen und Prozedurnamen unter Mitwirkung der Routine `ab_fileext_prozedur_zu_ziel` ermittelt werden. Als Eingabe werden eine Zieldateiextension und ein Prozedurname erwartet. Beide Angaben müssen nicht gemacht werden. Allerdings muß eine Angabe getätigt werden (entweder über Zieldateiextension oder über Prozedurnamen). Nach richtiger Angabe gibt es folgende Unterscheidungsfälle:

- Angabe sowohl über Zieldateiextension als auch über Prozedurnamen:
In diesem Fall kann überprüft werden, ob die Zuordnung zwischen der Zieldateiextension und der Prozedur in der Datenbank vorhanden ist.
- Nur Angabe über Dateiextension:
In diesem Fall kann bestimmt werden, zu welchen Prozeduren die angegebene Zieldateiextension eine Beziehung eingeht. Mit anderem Wort, es wird ermittelt, welche Prozeduren auf die Sprachsignaldatei mit dieser Zieldateiextension angewandt werden können.

- Nur Angabe über Prozedurnamen:
In diesem Fall wird ermittelt, mit welcher Zieldateiextension eine Prozedur assoziiert ist.

Option A=1B=2C=4

Zusammenfassung:

Eingabe: Zieldateiextension und Prozedurnamen

oder

Zieldateiextension

oder

Prozedurnamen

Automatisierungsmodus:

korpo -s "Datenbankname" A=1 B=2 C=4 "Zieldateiextension"

"Prozedurnamen" -e

oder

korpo -s "Datenbankname" A=1 B=2 C=4 "Zieldateiextension" -e

oder

korpo -s "Datenbankname" A=1 B=2 C=4 "Prozedurnamen" -e

• **5: Prozedurauswahl nach Namen**

Bei dieser Auswahl kann man unter Angabe eines Prozedurnamens mit Hilfe der Routine `ab_prozedur_name_auswahl` herausfinden, ob eine bestimmte Prozedur in der Datenbank vorkommt, die durch den angegebenen Namen identifiziert wird.

Option A=1B=2C=5

Zusammenfassung:

Eingabe: Prozedurnamen

Automatisierungsmodus:

korpo -s "Datenbankname" A=1 B=2 C=5 "Prozedurnamen" -e

• **6: Prozedurauswahl nach Typen**

Bei dieser Auswahl kann man unter Angabe eines Prozedurtyps mit Hilfe der Routine `ab_prozedur_typ_auswahl` herausfinden, ob eine bestimmte Prozedur in der Datenbank vorkommt.

Option A=1B=2C=6

Zusammenfassung:

Eingabe: Prozedurtype

Automatisierungsmodus:

korpo -s "Datenbankname" A=1 B=2 C=6 "Prozedurtyp" -e

3.4.9 Menü A1B3_C

- **1: Alle Formate**

Mit Hilfe dieses Menüpunkts können alle vorhandenen Datensätze der Tabelle FORMAT aufgelistet werden. Die dabei verwendete Routine ist `ab_format_alle`. Keine Eingabe wird erwartet.

Option `A=1B=3C=1`

Zusammenfassung:

Eingabe: Keine

Automatisierungsmodus:

`korpo -s "Datenbankname" A=1 B=3 C=1 -e`

- **2: Zugeordnete Formate**

Bei dieser Auswahl werden Format-Elemente unter Mitwirkung der Routine `ab_format_zugeordnet` ermittelt, die einer Sprachsignaldatei zugeordnet sind. Es werden zunächst alle Datensätze aus der Tabelle FORMAT ausgelesen, bei denen einzeln untersucht wird, ob das vorliegende Format einer Sprachsignaldatei zugewiesen wird. Die Zuordnung wird bekanntlich durch einen Datensatz in der Tabelle `FILE_FO_ZUORDNUNG` ausgedrückt, in dem das betroffene Format vorkommt.

Option `A=1B=3C=2`

Zusammenfassung:

Eingabe: Keine

Automatisierungsmodus:

`korpo -s "Datenbankname" A=1 B=3 C=2 -e`

- **3: Formatauswahl nach Kennzeichnung**

Bei dieser Auswahl kann man unter Angabe einer Formatkennzeichnung mit Hilfe der Routine `ab_format_name_auswahl` herausfinden, ob ein bestimmtes Format in der Datenbank vorkommt.

Option `A=1B=3C=3`

Zusammenfassung:

Eingabe: Formatkennzeichnung (obligatorisch)

Automatisierungsmodus:

`korpo -s "Datenbankname" A=1 B=3 C=3 "Formatkennzeichnung" -e`

3.4.10 Menü A1B4_C

- **1: Alle Haupttypen**

Mit Hilfe dieses Menüpunkts können alle vorhandenen Datensätze der Tabelle HAUPTTYP aufgelistet werden. Die dabei verwendete Routine ist `ab_haupttyp_alle`. Keine Eingabe wird erwartet.

Option A=1B=4C=1

Zusammenfassung:

Eingabe: Keine

Automatisierungsmodus:

korpo -s "Datenbankname" A=1 B=4 C=1 -e

- **2: Zugeordnete Haupttypen**

Bei dieser Auswahl werden Haupttyp-Elemente unter Mitwirkung der Routine `ab_haupttyp_zugeordnet` ermittelt, die einer Sprachsignaldatei zugeordnet sind. Es werden zunächst alle Datensätze aus der Tabelle HAUPTTYP ausgelesen, bei denen einzeln untersucht wird, ob der vorliegende Haupttyp einer Sprachsignaldatei zugewiesen wird. Die Zuordnung wird bekanntlich durch einen Datensatz in der Tabelle FILE_HT_ZUORDNUNG ausgedrückt, in dem der betroffene Haupttyp partizipiert ist.

Option A=1B=4C=2

Zusammenfassung:

Eingabe: Keine

Automatisierungsmodus:

korpo -s "Datenbankname" A=1 B=4 C=2 -e

- **3: Haupttypauswahl nach Kennzeichnung**

Bei dieser Auswahl kann man unter Angabe einer Haupttypkennzeichnung mit Hilfe der Routine `ab_haupttyp_name_auswahl` herausfinden, ob ein bestimmter Haupttyp in der Datenbank vorkommt.

Option A=1B=4C=3

Zusammenfassung:

Eingabe: Haupttypkennzeichnung (obligatorisch)

Automatisierungsmodus:

korpo -s "Datenbankname" A=1 B=4 C=3 "Haupttypkennzeichnung" -e

3.4.11 Menü A1B5_C

- **1: Alle Untertypen**

Mit Hilfe dieses Menüpunkts können alle vorhandenen Datensätze der Tabelle UNTERTYP aufgelistet werden. Die dabei verwendete Routine ist `ab_untertyp_alle`. Keine Eingabe wird erwartet.

Option A=1B=5C=1

Zusammenfassung:

Eingabe: Keine

Automatisierungsmodus:

`korpo -s "Datenbankname" A=1 B=5 C=1 -e`

• **2: Zugeordnete Untertypen**

Bei dieser Auswahl werden Untertyp-Elemente unter Mitwirkung der Routine `ab_untertyp_zugeordnet` ermittelt, die einer Sprachsignaldatei zugeordnet sind. Es werden zunächst alle Datensätze aus der Tabelle UNTERTYP ausgelesen, bei denen einzeln untersucht wird, ob der vorliegende Untertyp einer Sprachsignaldatei zugewiesen wird. Die Zuordnung wird bekanntlich durch einen Datensatz in der Tabelle FILE_UT_ZUORDNUNG ausgedrückt, in dem der betroffene Untertyp partizipiert ist.

Option A=1B=5C=2

Zusammenfassung:

Eingabe: Keine

Automatisierungsmodus:

`korpo -s "Datenbankname" A=1 B=5 C=2 -e`

• **3: Untertypauswahl nach Kennzeichnung**

Bei dieser Auswahl kann man unter Angabe einer Untertypkennzeichnung mit Hilfe der Routine `ab_untertyp_name_auswahl` herausfinden, ob ein bestimmter Untertyp in der Datenbank vorkommt.

Option A=1B=5C=3

Zusammenfassung:

Eingabe: Untertypkennzeichnung (obligatorisch)

Automatisierungsmodus:

`korpo -s "Datenbankname" A=1 B=5 C=3 "Untertypkennzeichnung" -e`

3.4.12 Menü A1B6_C

• **1: Alle Quellen**

Mit Hilfe dieses Menüpunkts können alle vorhandenen Datensätze der Tabelle QUELLE aufgelistet werden. Die dabei verwendete Routine ist `ab_quelle_alle`. Keine Eingabe wird erwartet.

Option A=1B=6C=1

Zusammenfassung:

Eingabe: Keine

Automatisierungsmodus:

korpo -s "Datenbankname" A=1 B=6 C=1 -e

- **2: Zugeordnete Quellen**

Bei dieser Auswahl werden unter Mitwirkung der Routine `ab_quelle_zugeordnet` diejenigen Quellen ermittelt, von denen eine Sprachsignaldatei stammen kann. Es werden zunächst alle Datensätze aus der Tabelle `QUELLE` ausgelesen, bei denen einzeln untersucht wird, ob die vorliegende Quelle einer Sprachsignaldatei zugewiesen wird. Die Zuordnung wird bekanntlich durch einen Datensatz in der Tabelle `FILE_QU_ZUORDNUNG` ausgedrückt, in dem die betroffene Quelle vorkommt. Es ist hierbei angebracht, anzumerken, daß die Quelleinformationen in der Tabelle `FILE_QU_ZUORDNUNG` u.U. redundant sind, weil Informationen über die Quelle einer Sprachsignaldatei auch im Feld "Quelle" in der Tabelle `FILE` vorhanden sind. Trotzdem ist die Tabelle hilfreich beim Zugriff auf Quellen von Sprachsignaldateien und kann als Zusatzinformation betrachtet werden, sollte die zugeordnete Quelle einer Sprachsignaldatei in der Tabelle `FILE` aus welchen Gründen auch immer fehlen.

Option A=1B=6C=2

Zusammenfassung:

Eingabe: Keine

Automatisierungsmodus:

korpo -s "Datenbankname" A=1 B=6 C=2 -e

- **3: Quelle-Auswahl nach Namen**

Bei dieser Auswahl kann man unter Angabe von Quellebezeichnungen mit Hilfe der Routine `ab_quelle_name_auswahl` herausfinden, ob eine bestimmter Quelle in der Datenbank vorkommt.

Option A=1B=6C=3

Zusammenfassung:

Eingabe: Bezeichnung 1 (obligatorisch), Bezeichnung 2 (optional)

Automatisierungsmodus:

korpo -s "Datenbankname" A=1 B=6 C=3 "Bezeichnung 1" "Bezeichnung 2" -e
oder

korpo -s "Datenbankname" A=1 B=6 C=3 "Bezeichnung 1" -e

3.4.13 Menü A1B7_C

- **1: Alle Zugriffsrechte**

Mit Hilfe dieses Menüpunkts können alle vorhandenen Datensätze der Tabelle ZUGRIFFSRECHT aufgelistet werden. Die dabei verwendete Routine ist `ab_zugriff_alle`. Keine Eingabe wird erwartet.

Option A=1B=7C=1

Zusammenfassung:

Eingabe: Keine

Automatisierungsmodus:

`korpo -s "Datenbankname" A=1 B=7 C=1 -e`

• **2: Zugeordnete Zugriffsrechte mit Dateien**

Bei dieser Auswahl werden Zugriffsrechte unter Mitwirkung der Routine `ab_zugriff_zugeordnet` ermittelt, die an eine Sprachsignaldatei vergeben sind. Es werden zunächst alle Datensätze aus der Tabelle ZUGRIFFSRECHT ausgelesen, bei denen einzeln untersucht wird, ob das vorliegende Zugriffsrecht einer Sprachsignaldatei zugewiesen wird. Die Zuordnung wird bekanntlich durch einen Datensatz in der Tabelle FILE_ZUG_ZUORDNUNG ausgedrückt, in dem das betroffene Zugriffsrecht partizipiert ist.

Option A=1B=7C=2

Zusammenfassung:

Eingabe: Keine

Automatisierungsmodus:

`korpo -s "Datenbankname" A=1 B=7 C=2 -e`

3.4.14 Menü A3B1_C

• **1: Interaktiv**

Mit diesem Menüpunkt kann der Benutzer auf interaktive Weise Daten in die Tabelle FILE eintragen. Dies geschieht mit Hilfe der Routine `db_file_eintragen`. Innerhalb dieser Routine wird eine unendliche Schleife gebildet, in der benötigte Informationen über die einzutragenden Datensätze angegeben werden können. Obligatorische Angaben sind: Dateiname, Dateiextension und Dateipfad. Optionale Angaben sind: Stammdaten, Datum, Uhrzeit und Quelle. Vor jeder Eintragung von Daten in die Tabelle wird immer zuerst überprüft, ob der vorliegende Datensatz bereits in der Datenbank vorhanden ist. Somit wird gewährleistet, daß die vorhandenen Daten immer eindeutig sind. Der Benutzer wird darüber informiert, soll das Eintragen von Daten fehlschlagen.

Option A=3B=1C=1

Zusammenfassung:

Eingabe: Dateiname (obligatorisch)

Stammdaten (optional)

Datum (optional)

Uhrzeit (optional)
 Dateiextension (obligatorisch)
 Dateipfad (obligatorisch)
 Dateiquelle (optional)

Automatisierungsmodus:

```
korpo -s "Datenbankname" A=3 B=1 C=1 "Dateinamen" "Dateiextension"
"Dateipfad" "Dateistamm" "Datum" "Uhrzeit" "Quelle" -e
oder
korpo -s "Datenbankname" A=3 B=1 C=1 "Dateinamen" "Dateiextension"
"Dateipfad" -e
```

Dabei ist es zu beachten, daß im Automatisierungsmodus die gleiche Bedingung gilt in Bezug auf die Beschaffenheit der Argumente, deren Anzahl und deren Reihenfolge - der Dateiname, die Dateiextension und der Dateipfad sind nämlich ebenfalls obligatorisch und müssen in dieser Reihenfolge angegeben werden. Alle anderen Argumente sind optional.

• 2: Einlesen von Dateien

Analog zum Löschen von Daten aus der Tabelle FILE (siehe 3.4.5 in Batchform können hierbei Datensätze aus einer In putdatei mit Hilfe der Routine db_file_einlesen_haupt in dieselbe Tabelle eingetragen werden. Diese Routine wird auch verwendet beim Löschvorgang, der Unterschied besteht diesmal nur darin, daß ihr andere Argumente übergeben werden. Konkret bedeutet dies:

- Operationskennung lautet nun "r" ("d" für Löschen).
- Die Funktion db_file_read_file wird als 2. Argument übergeben.

Ansonsten zeigt das Programm in Bezug auf die Routine db_file_einlesen_haupt hier einen ähnlichen Ablauf wie beim Löschen. Die Inputdatei wird auch ähnlich aufgebaut, die für das Einlesen zuständige Routine db_file_read_file prüft ebenfalls zuers t, ob die Inputdatei bezüglich der Konfigurationsdaten in Ordnung ist. Soll es der Fall sein, wird die Inputdatei Zeile für Zeile abgearbeitet.

Option A=3B=1C=2

Zusammenfassung:

Eingabe: Inputdatei (obligatorisch)

Automatisierungsmodus:

```
korpo -s "Datenbankname" A=3 B=1 C=2 "Inputdatei" -e
```

3.4.15 Menü A4B8_C

• 1: Datei-Format-Zuordnung löschen

Mit Hilfe dieses Menüpunkt können Datensätze aus der Tabelle FILE_FO_ZUORDNUNG gelöscht werden. Hierzu wird die Routine db_file_format_zuordnung aufgerufen, die als erstes Angabe über einen Dateinamen entgegennimmt. Diese Angabe muß auf jeden Fall gemacht werden, bevor es weiter gehen kann. Danach wird überprüft, ob die angegebene Datei in der Tabelle FILE zu finden ist. Im positiven Fall wird der Benutzer aufgefordert, eine

Formatkennzeichnung einzugeben. Angabe über Formatkennzeichnung ist jedoch nicht obligatorisch. Es werden anschließend zwei Fälle unterschieden:

- Formatkennzeichnung ist nicht eingegeben:
In diesem Fall wird eine Rückfrage an den Benutzer gestellt, indem der Benutzer bestätigen muß, ob alle Datensätze aus der Tabelle FILE_FO_ZUORDNUNG gelöscht werden sollen, die den angegebenen Dateinamen besitzen. Bei Ja-Antwort werden all die qualifizierten Datensätze gelöscht.
- Formatkennzeichnung ist eingegeben:
In diesem Fall werden nur diejenige Datensätze gelöscht, die den Angaben voll und ganz genügen.

Option A=4B=8C=1

Zusammenfassung:

Eingabe: Dateiname (obligatorisch)

Formatkennzeichnung (optional)

Automatisierungsmodus:

korpo -s "Datenbankname" A=4 B=8 C=1 "Dateinamen"

"Formatkennzeichnung" -e

oder

korpo -s "Datenbankname" A=4 B=8 C=1 "Dateinamen" -e

• **2: Datei-Prozedur-Zuordnung löschen**

Dieser Menüpunkt wird zu Hilfe gezogen, sollen Datensätzen aus der Tabelle FILE_PRO_ZUORDNUNG gelöscht werden. Dabei wird die Routine db_file_prozedur_zuordnung_loeschen aktiviert. Auch in diesem Fall muß als erstes ein Dateiname nach Aufforderung eingegeben werden. Es wird auf jeden Fall festgestellt, ob diese Datei in der Datenbank vorkommt. Soll es der Fall sein, wird ein Prozedurname erwartet, der allerdings nicht unbedingt anzugeben ist (die Aufforderung kann mit *< RETURN >* quittiert werden). Ist die Angabe in Ordnung, kommen wiederum zwei Unterscheidungsfälle in Betracht:

- Prozedurname fehlt:
In diesem Fall werden all die Datensätze aus der Tabelle FILE_PRO_ZUORDNUNG gelöscht, die den angegebenen Dateinamen haben. Zur Sicherheit wird eine Rückfrage gestellt, die mit "Ja" quittiert werden muß, bevor die Daten endlich gelöscht werden.
- Prozedurname vorhanden:
In diesem Fall werden nur diejenige Datensätze gelöscht, die den Angaben voll und ganz genügen.

Option A=4B=8C=2

Zusammenfassung:

Eingabe: Dateiname (obligatorisch)

Prozedurname (optional)

Automatisierungsmodus:

korpo -s "Datenbankname" A=4 B=8 C=2 "Dateinamen"

"Prozedurnamen" -e

oder

korpo -s "Datenbankname" A=4 B=8 C=2 "Dateinamen" -e

- **3: Datei-Haupttyp-Zuordnung löschen**

Dieser Menüpunkt wird zu Hilfe gezogen, sollen Datensätzen aus der Tabelle FILE_HT_ZUORDNUNG gelöscht werden. Dabei wird die Routine db_file_haupttyp_zuordnung_loeschen aktiviert. Auch in diesem Fall muß als erstes ein Dateiname nach Aufforderung eingegeben werden. Es wird auf jeden Fall festgestellt, ob diese Datei in der Datenbank vorkommt. Soll es der Fall sein, wird ein Haupttyp-Kennzeichnung erwartet, der allerdings nicht unbedingt anzugeben ist (die Aufforderung kann mit *< RETURN >* quittiert werden). Ist die Angabe in Ordnung, kommen wiederum zwei Unterscheidungsfälle in Betracht:

- Haupttypkennzeichnung fehlt:
In diesem Fall werden all die Datensätze aus der Tabelle FILE_HT_ZUORDNUNG gelöscht, die den angegebenen Dateinamen haben. Zur Sicherheit wird eine Rückfrage gestellt, die mit "Ja" quittiert werden muß, bevor die Daten endlich gelöscht werden.
- Haupttypkennzeichnung vorhanden:
In diesem Fall werden nur diejenige Datensätze gelöscht, die den Angaben voll und ganz genügen.

Option A=4B=8C=3

Zusammenfassung:

Eingabe: Dateiname (obligatorisch)

Haupttypkennzeichnung (optional)

Automatisierungsmodus:

korpo -s "Datenbankname" A=4 B=8 C=3 "Dateinamen"

"Haupttypkennzeichnung" -e

oder

korpo -s "Datenbankname" A=4 B=8 C=3 "Dateinamen" -e

- **4: Datei-Untertyp-Zuordnung löschen**

Dieser Menüpunkt wird zu Hilfe gezogen, sollen Datensätzen aus der Tabelle FILE_UT_ZUORDNUNG gelöscht werden. Dabei wird die Routine db_file_untertyp_zuordnung_loeschen aktiviert. Auch in diesem Fall muß als erstes ein Dateiname nach Aufforderung eingegeben werden. Es wird auf jeden Fall festgestellt, ob diese Datei in der Datenbank vorkommt. Soll es der Fall sein, wird ein Untertypkennzeichnung erwartet, der allerdings nicht unbedingt anzugeben ist (die Aufforderung kann mit *< RETURN >* quittiert werden). Ist die Angabe in Ordnung, kommen wiederum zwei Unterscheidungsfälle in Betracht:

- Untertypkennzeichnung fehlt:
In diesem Fall werden all die Datensätze aus der Tabelle FILE_UT_ZUORDNUNG gelöscht, die den angegebenen Dateinamen haben. Zur Sicherheit wird eine Rückfrage gestellt, die mit "Ja" quittiert werden muß, bevor die Daten endlich gelöscht werden.
- Untertypkennzeichnung vorhanden:
In diesem Fall werden nur diejenige Datensätze gelöscht, die den Angaben voll und ganz genügen.

Option A=4B=8C=4

Zusammenfassung:

Eingabe: Dateiname (obligatorisch)

Untertypkennzeichnung (optional)

Automatisierungsmodus:

korpo -s "Datenbankname" A=4 B=8 C=4 "Dateinamen"

"Untertypkennzeichnung" -e

oder

korpo -s "Datenbankname" A=4 B=8 C=4 "Dateinamen" -e

• 5: Datei-Quelle-Zuordnung löschen

Dieser Menüpunkt wird zu Hilfe gezogen, sollen Datensätzen aus der Tabelle FILE_QU_ZUORDNUNG gelöscht werden. Dabei wird die Routine db_file_quelle_zuordnung_loeschen aktiviert. Auch in diesem Fall muß als erstes ein Dateiname nach Aufforderung eingegeben werden. Es wird auf jeden Fall festgestellt, ob diese Datei in der Datenbank vorkommt. Soll es der Fall sein, wird die 1. Bezeichnung der Quelle erwartet, die allerdings nicht unbedingt anzugeben ist (die Aufforderung kann mit *< RETURN >* quittiert werden). Ist die Angabe in Ordnung, kommen wiederum zwei Unterscheidungsfälle in Betracht:

- 1. Bezeichnung der Quelle fehlt:

In diesem Fall werden all die Datensätze aus der Tabelle FILE_QU_ZUORDNUNG gelöscht, die den angegebenen Dateinamen haben. Zur Sicherheit wird eine Rückfrage gestellt, die mit "Ja" quittiert werden muß, bevor die Daten endlich gelöscht werden.

- 1. Bezeichnung der Quelle vorhanden:

In diesem Fall werden nur diejenige Datensätze gelöscht, die den Angaben voll und ganz genügen.

Option A=4B=8C=5

Zusammenfassung:

Eingabe: Dateiname (obligatorisch)

1. Bezeichnung der Quelle (optional)

Automatisierungsmodus:

korpo -s "Datenbankname" A=4 B=8 C=5 "Dateinamen"

"Quellebezeichnung" -e

oder

korpo -s "Datenbankname" A=4 B=8 C=5 "Dateinamen" -e

• 6: Datei-Zugriff-Zuordnung löschen

Dieser Menüpunkt wird zu Hilfe gezogen, sollen Datensätzen aus der Tabelle FILE_ZUG_ZUORDNUNG gelöscht werden. Dabei wird die Routine db_file_zugriff_zuordnung_loeschen aktiviert. Auch in diesem Fall muß als erstes ein Dateiname nach Aufforderung eingegeben werden. Es wird auf jeden Fall festgestellt, ob diese Datei in der Datenbank vorkommt. Soll es der Fall sein, wird eine Zugriffsrecht-ID erwartet, die allerdings nicht unbedingt anzugeben ist (die Aufforderung kann mit *< RETURN >* quittiert werden). Ist die Angabe in Ordnung, kommen wiederum zwei Unterscheidungsfälle in Betracht:

- Zugriffsrecht-ID fehlt:
In diesem Fall werden all die Datensätze aus der Tabelle FILE_ZUG_ZUORDNUNG gelöscht, die den angegebenen Dateinamen haben. Zur Sicherheit wird eine Rückfrage gestellt, die mit "Ja" quittiert werden muß, bevor die Daten endlich gelöscht werden.
- Zugriffsrecht-ID vorhanden:
In diesem Fall werden nur diejenige Datensätze gelöscht, die den Angaben voll und ganz genügen.

Option A=4B=8C=6

Zusammenfassung:

Eingabe: Dateiname (obligatorisch)

Zugriffsrecht-ID (optional)

Automatisierungsmodus:

korpo -s "Datenbankname" A=4 B=8 C=6 "Dateinamen"

"Zugriffsrecht-ID" -e

oder

korpo -s "Datenbankname" A=4 B=8 C=6 "Dateinamen" -e

3.4.16 Menü A5B1_C

- **1: Datei-Änderungen-Allgemein**

Dieser Menüpunkt ist hilfreich für Änderungsoperationen, die auf Sprachsignaldateien aus der Tabelle FILE angewandt werden können, und bedient sich der Routine db_change_file. Da jede Sprachsignaldatei durch ihren Namen eindeutig identifiziert wird, wird zuerst ein Dateiname erwartet. Anschließend wird der Benutzer aufgefordert, einen neuen Dateinamen einzugeben. Außerdem können Angaben über Stammdaten, Datum, Dateiextension, Uhrzeit, Pfad und Quelle gemacht werden. Alle Angaben sind hierbei optional mit Ausnahme von dem Dateinamen, der die zu ändernde Sprachsignaldatei eindeutig kennzeichnet. Es soll hier betont werden, daß man bei diesem Menüpunkt die Möglichkeit hat, auf einmal Änderungen an beliebigen Feldern der Tabelle FILE durchzuführen, unter der Voraussetzung, daß neue, angebrachte Daten eingegeben werden. Vor der Änderung wird noch überprüft, ob die zu ändernde Sprachsignaldatei in der Datenbank vorhanden ist oder nicht. Ist die Prüfung in Ordnung, kann die gewünschte Änderung ausgeführt werden.

Option A=5B=1C=1

Zusammenfassung:

Eingabe: identifizierender Dateiname (obligatorisch)

neuer Dateiname (optional)

neuer Stamm (optional)

neues Datum (optional)

neue Uhrzeit (optional)

neue Dateiextension (optional)

neuer Pfad (optional)

neue Quelle (optional)

Automatisierungsmodus:

```
korpo -s "Datenbankname" A=5 B=1 C=1 "Dateinamen"
"neuer Dateiname" "neuer Stamm" "neues Datum"
"neue Uhrzeit" "neue Extension" "neuer Pfad"
"neue Quelle" -e
oder
korpo -s "Datenbankname" A=5 B=1 C=1 "Dateinamen"
"neuer Dateiname" "neuer Pfad" -e
oder
korpo -s "Datenbankname" A=5 B=1 C=1 "Dateinamen"
"neuer Stamm" "neue Quelle" -e
....
```

Analog zur Eintragung von Daten in die Tabelle FILE (siehe 3.4.14) , gilt im Automatisierungsmodus die gleiche Bedingung in Bezug auf die B eschaffenheit der Argumente, deren Anzahl und deren Reihenfolge.

• 2: Datei-Änderungen-Namen

Bei diese Auswahl kann der Name einer Sprachsignaldatei geändert werden. Die Hilfsroutine heißt in diesem Fall `db_change_file_namen`. Zwei Eingaben werden erwartet:

- der identifizierende Dateiname
- der neue Dateiname

Fehlt die Angabe über den neuen Dateinamen, kehrt das Programm zurück. Auch hier wird überprüft, ob die zu ändernde Sprachsignaldatei in der Datenbank vorkommt. Soll es der Fall sein, wird die Änderung vorgenommen.

Option A=5B=1C=2

Zusammenfassung:

Eingabe: identifizierender Dateiname (obligatorisch)
 neuer Dateiname (obligatorisch)

Automatisierungsmodus:

```
korpo -s "Datenbankname" A=5 B=1 C=2 "Dateinamen"
"neuen Dateinamen" -e
oder
korpo -s "Datenbankname" A=5 B=1 C=2 "Dateinamen" -e
```

• 3: Datei-Änderungen-Stamm

Bei diese Auswahl kann der Stamm einer Sprachsignaldatei geändert werden. Die Hilfsroutine heißt in diesem Fall `db_change_file_stamm`. Zwei Eingaben werden erwartet:

- der identifizierende Dateiname
- der neue Stamm

Fehlt die Angabe über den neuen Stamm, kehrt das Programm zurück. Auch hier wird überprüft, ob die zu ändernde Sprachsignaldatei in der Datenbank vorkommt. Soll es der Fall sein, wird die Änderung vorgenommen.

Option A=5B=1C=3

Zusammenfassung:

Eingabe: identifizierender Dateiname (obligatorisch)
 neuer Stamm (optional)

Automatisierungsmodus:

korpo -s "Datenbankname" A=5 B=1 C=3 "Dateinamen"
 "neuen Stamm" -e

oder

korpo -s "Datenbankname" A=5 B=1 C=3 "Dateinamen" -e

- **4: Datei-Änderungen-Datum**

Bei diese Auswahl kann das zugehörige Datum einer Sprachsignaldatei geändert werden. Die Hilfsroutine heißt in diesem Fall `db_change_file_datum`. Zwei Eingaben werden erwartet:

- der identifizierende Dateiname
- das neue Datum

Fehlt die Angabe über das neue Datum, kehrt das Programm zurück. Auch hier wird überprüft, ob die zu ändernde Sprachsignaldatei in der Datenbank vorkommt. Soll es der Fall sein, wird die Änderung vorgenommen.

Option A=5B=1C=4

Zusammenfassung:

Eingabe: identifizierender Dateiname (obligatorisch)
 neues Datum (optional)

Automatisierungsmodus:

korpo -s "Datenbankname" A=5 B=1 C=4 "Dateinamen"
 "neues Datum" -e

oder

korpo -s "Datenbankname" A=5 B=1 C=4 "Dateinamen" -e

- **5: Datei-Änderungen-Uhrzeit**

Bei diese Auswahl kann die zugehörige Uhrzeit einer Sprachsignaldatei geändert werden. Die Hilfsroutine heißt in diesem Fall `db_change_file_uhrzeit`. Zwei Eingaben werden erwartet:

- der identifizierende Dateiname
- die neue Uhrzeit

Fehlt die Angabe über die neue Uhrzeit, kehrt das Programm zurück. Auch hier wird überprüft, ob die zu ändernde Sprachsignaldatei in der Datenbank vorkommt. Soll es der Fall sein, wird die Änderung vorgenommen.

Option A=5B=1C=5

Zusammenfassung:

Eingabe: identifizierender Dateiname (obligatorisch)
 neue Uhrzeit (optional)

Automatisierungsmodus:

```
korpo -s "Datenbankname" A=5 B=1 C=5 "Dateinamen"
```

```
"neue Uhrzeit" -e
```

oder

```
korpo -s "Datenbankname" A=5 B=1 C=5 "Dateinamen" -e
```

• 6: Datei-Änderungen-Extension

Bei diese Auswahl kann die Dateiextension einer Sprachsignaldatei geändert werden. Die Hilfsroutine heißt in diesem Fall `db_change_file_endung`. Zwei Eingaben werden erwartet:

- der identifizierende Dateiname
- die neue Dateiextension

Fehlt die Angabe über die neue Dateiextension, kehrt das Programm zurück. Auch hier wird überprüft, ob die zu ändernde Sprachsignaldatei in der Datenbank vorkommt. Soll es der Fall sein, wird die Änderung vorgenommen.

Option A=5B=1C=6

Zusammenfassung:

Eingabe: identifizierender Dateiname (obligatorisch)

neue Dateiextension (optional)

Automatisierungsmodus:

```
korpo -s "Datenbankname" A=5 B=1 C=6 "Dateinamen"
```

```
"neue Dateiextension" -e
```

oder

```
korpo -s "Datenbankname" A=5 B=1 C=6 "Dateinamen" -e
```

• 7: Datei-Änderungen-Pfad

Bei diese Auswahl kann der Pfad einer Sprachsignaldatei geändert werden. Die Hilfsroutine heißt in diesem Fall `db_change_file_pfad`. Zwei Eingaben werden erwartet:

- der identifizierende Dateiname
- der neue Pfad

Fehlt die Angabe über den neuen Pfad, kehrt das Programm zurück. Auch hier wird überprüft, ob die zu ändernde Sprachsignaldatei in der Datenbank vorkommt. Soll es der Fall sein, wird die Änderung vorgenommen.

Option A=5B=1C=7

Zusammenfassung:

Eingabe: identifizierender Dateiname (obligatorisch)

neuen Pfad (optional)

Automatisierungsmodus:

```
korpo -s "Datenbankname" A=5 B=1 C=7 "Dateinamen"
```

```
"neuen Pfad" -e
```

oder

```
korpo -s "Datenbankname" A=5 B=1 C=7 "Dateinamen" -e
```

- **8: Datei-Änderungen-Quelle**

Bei diese Auswahl kann die Quelle einer Sprachsignaldatei geändert werden. Die Hilfsroutine heißt in diesem Fall `db_change_file_quelle`. Zwei Eingaben werden erwartet:

- der identifizierende Dateiname
- die neue Quelle

Fehlt die Angabe über die neue Quelle, kehrt das Programm zurück. Auch hier wird überprüft, ob die zu ändernde Sprachsignaldatei in der Datenbank vorkommt. Soll es der Fall sein, wird die Änderung vorgenommen.

Option `A=5B=1C=8`

Zusammenfassung:

Eingabe: identifizierender Dateiname (obligatorisch)
neue Quelle (optional)

Automatisierungsmodus:

`korpo -s "Datenbankname" A=5 B=1 C=8 "Dateinamen"`
`"neue Quelle" -e`

oder

`korpo -s "Datenbankname" A=5 B=1 C=8 "Dateinamen" -e`

3.4.17 Menü `A5B3_C`

- **1: Prozedur-Änderung-Allgemein**

Dieser Menüpunkt ermöglicht, Änderungen in den Datenbeständen der Tabelle `PROZEDUR` an mehreren Feldern vorzunehmen. Dabei wird die Routine `db_change_prozedur` aufgerufen, die wie folgt verläuft:

Aufforderung zur Eingabe eines Prozedurnamens, der den zu ändernden Datensatz eindeutig identifiziert.

Aufforderung zur möglichen Eingabe eines neuen Prozedurnamens.

Aufforderung zur möglichen Eingabe eines neuen Prozedurtyps.

Soll die Angabe von erforderlichen Daten in Ordnung sein, wird als nächstes überprüft, ob die zu ändernde Prozedur in der Tat in der Datenbank vorhanden ist. Soll es der Fall sein und bei entsprechender Dateneingabe wird die Änderung vorgenommen.

Option `A=5B=3C=1`

Zusammenfassung:

Eingabe: identifizierender Prozedurname (obligatorisch)
neuer Prozedurname (obligatorisch)
neuer Prozedurtyp (optional)

Automatisierungsmodus:

`korpo -s "Datenbankname" A=5 B=3 C=1 "Prozedurnamen"`
`"neuen Prozedurnamen" "neuen Prozedurtyp" -e`

oder

- **2: Prozedur-Änderung-Namen**

Bei dieser Auswahl bekommt man ein Werkzeug zur Hand, mit dem Prozedurnamen aus der Tabelle PROZEDUR geändert werden sollen. Hierzu wird die Routine `db_change_prozedur_namen` zu Hilfe gezogen. Als erstes muß der identifizierende Prozedurname eingegeben werden, dann folgt die Angabe über den neuen Prozedurnamen, die allerdings optional ist. Fehlt diese Angabe, findet die Änderungsoperation gar nicht statt. Vor der Änderung wird überprüft, ob die zu ändernde Prozedur überhaupt in der Datenbank existiert. Soll es der Fall sein, wird erst recht die Änderung vorgenommen.

Option A=5B=3C=2

Zusammenfassung:

Eingabe: identifizierender Prozedurname (obligatorisch)
 neuer Prozedurname (optional)

Automatisierungsmodus:

`korpo -s "Datenbankname" A=5 B=3 C=2 "Prozedurnamen"`
`"neuen Prozedurnamen" -e`
 oder

- **3: Prozedur-Änderung-Typ**

Bei dieser Auswahl bekommt man ein Werkzeug zur Hand, mit dem Prozedurtypen aus der Tabelle PROZEDUR geändert werden sollen. Hierzu wird die Routine `db_change_prozedur_typ` zu Hilfe gezogen. Als erstes muß der identifizierende Prozedurname eingegeben werden, dann folgt die Angabe über den neuen Prozedurtyp, die allerdings optional ist. Fehlt diese Angabe, findet die Änderungsoperation gar nicht statt. Vor der Änderung wird überprüft, ob die zu ändernde Prozedur überhaupt in der Datenbank existiert. Soll es der Fall sein, wird erst recht die Änderung vorgenommen.

Option A=5B=3C=3

Zusammenfassung:

Eingabe: identifizierender Prozedurname (obligatorisch)
 neuer Prozedurtyp (optional)

Automatisierungsmodus:

`korpo -s "Datenbankname" A=5 B=3 C=3 "Prozedurnamen"`
`"neuen Prozedurtyp" -e`
 oder

3.4.18 Menü A5B7_C

- **1: Dateizuordnungen-Änderungen-Allgemein**

Bei dieser Auswahl kann man Änderungen an den Datensätzen derjenigen Tabellen durchführen, die eine Beziehung zu Sprachsignaldateien in der Tabelle FILE pflegen. In erster Line handelt es sich um die mehrmals erwähnten Tabellen

FILE_FO_ZUORDNUNG, FILE_PRO_ZUORDNUNG, FILE_HT_ZUORDNUNG, FILE_UT_ZUORDNUNG, FILE_QU_ZUORDNUNG und FILE_ZUG_ZUORDNUNG. Dieser Menüpunkt bedient sich der Routine `db_change_file_zuordnung`, die zuerst einen Dateinamen erwartet, welcher die Sprachsignaldatei fixiert, an deren Beziehung geändert werden soll. Anschließend wird überprüft, ob die angegebene Datei in der Datenbank vorhanden ist. Dabei wird der Identifikator dieser Sprachsignaldatei ermittelt (der Identifikator ist eine Integer-Zahl, und ist ein Datenfeld der Tabelle FILE). Ist die Überprüfung in Ordnung, dann werden der Reihe nach folgende Routinen aufgerufen, an die der ermittelte Identifikator als Parameter übergeben wird:

– `db_change_file_format_zuordnung_mID`:

Diese Routine ermöglicht die Änderung der Formatkennzeichnung, die der übergebenen Datei zugewiesen ist. Der Benutzer wird daraufhin aufgefordert, die zu ändernde Formatkennzeichnung und die neue Formatkennzeichnung einzugeben, wobei die erstgenannte optional ist. In Abhängigkeit von den Eingaben wird wie folgt unterschieden:

* zu ändernde Formatkennzeichnung nicht eingegeben:

In diesem Fall werden alle Datensätze in der Tabelle FILE_FO_ZUORDNUNG geändert, bei denen die übergebene Sprachsignaldatei vorkommt, ungeachtet der partizipierenden Formatkennzeichnung. Vor der Änderung muß aber die Bestätigung einer Rückfrage erfolgen.

* zu ändernde Formatkennzeichnung eingegeben:

In diesem Fall werden lediglich Datensätze aus der Tabelle FILE_FO_ZUORDNUNG geändert, die sowohl die übergebene Sprachsignaldatei als auch die zu ändernde Formatkennzeichnung enthalten.

Eingabe: zu ändernde Formatkennzeichnung
neue Formatkennzeichnung

– `db_change_file_prozedur_zuordnung_mID`:

Diese Routine ermöglicht die Änderung der Prozedur, die der übergebenen Datei zugewiesen ist. Der Benutzer wird daraufhin aufgefordert, den zu ändernden Prozedurnamen und den neuen Prozedurnamen einzugeben, wobei der erstgenannte optional ist. In Abhängigkeit von den Eingaben wird wie folgt unterschieden:

* zu ändernder Prozedurname nicht eingegeben:

In diesem Fall werden alle Datensätze in der Tabelle FILE_PRO_ZUORDNUNG geändert, bei denen die übergebene Sprachsignaldatei vorkommt, ungeachtet der partizipierenden Prozedur. Vor der Änderung muß aber die Bestätigung einer Rückfrage erfolgen.

* zu ändernder Prozedurname eingegeben:

In diesem Fall werden lediglich Datensätze aus der Tabelle FILE_PRO_ZUORDNUNG geändert, die sowohl die übergebene Sprachsignaldatei als auch den zu ändernden Prozedurnamen enthalten.

Eingabe: zu ändernder Prozedurname
neuer Prozedurname

– `db_change_file_haupttyp_zuordnung_mID`:

Diese Routine ermöglicht die Änderung des Haupttyps, der der übergebenen Datei zugewiesen ist. Der Benutzer wird daraufhin aufgefordert, den zu ändernden Haupttyp und den neuen Haupttyp einzugeben, wobei der erstgenannte optional ist. In Abhängigkeit von den Eingaben wird wie folgt unterschieden:

* zu ändernder Haupttyp nicht eingegeben:
In diesem Fall werden alle Datensätze in der Tabelle FILE_HT_ZUORDNUNG geändert, bei denen die übergebene Sprachsignaldatei vorkommt, ungeachtet des partizipierenden Haupttyps. Vor der Änderung muß aber die Bestätigung einer Rückfrage erfolgen.

* zu ändernder Haupttyp eingegeben:
In diesem Fall werden lediglich Datensätze aus der Tabelle FILE_HT_ZUORDNUNG geändert, die sowohl die übergebene Sprachsignaldatei als auch den zu ändernden Haupttyp enthalten.

Eingabe: zu ändernde Haupttypkennzeichnung
neue Haupttypkennzeichnung

– db_change_file_untertyp_zuordnung_mID:

Diese Routine ermöglicht die Änderung des Untertyps, der der übergebenen Datei zugewiesen ist. Der Benutzer wird daraufhin aufgefordert, den zu ändernden Untertyp und den neuen Untertyp einzugeben, wobei der erstgenannte optional ist. In Abhängigkeit von den Eingaben wird wie folgt unterschieden:

* zu ändernder Untertyp nicht eingegeben:
In diesem Fall werden alle Datensätze in der Tabelle FILE_UT_ZUORDNUNG geändert, bei denen die übergebene Sprachsignaldatei vorkommt, ungeachtet des partizipierenden Untertyps. Vor der Änderung muß aber die Bestätigung einer Rückfrage erfolgen.

* zu ändernder Untertyp eingegeben:
In diesem Fall werden lediglich Datensätze aus der Tabelle FILE_UT_ZUORDNUNG geändert, die sowohl die übergebene Sprachsignaldatei als auch den zu ändernden Untertyp enthalten.

Eingabe: zu ändernde Untertypkennzeichnung
neue Untertypkennzeichnung

– db_change_file_quelle_zuordnung_mID:

Diese Routine ermöglicht die Änderung der Quelle, die der übergebenen Datei zugewiesen ist. Der Benutzer wird daraufhin aufgefordert, die zu ändernde Quelle und die neue Quelle einzugeben, wobei die erstgenannte optional ist. In Abhängigkeit von den Eingaben wird wie folgt unterschieden:

* zu ändernde Quelle nicht eingegeben:
In diesem Fall werden alle Datensätze in der Tabelle FILE_QU_ZUORDNUNG geändert, bei denen die übergebene Sprachsignaldatei vorkommt, ungeachtet der partizipierenden Quelle. Vor der Änderung muß aber die Bestätigung einer Rückfrage erfolgen.

* zu ändernde Quelle eingegeben:
In diesem Fall werden lediglich Datensätze aus der Tabelle FILE_QU_ZUORDNUNG geändert, die sowohl die übergebene Sprachsignaldatei als auch die zu ändernde Quelle enthalten.

Eingabe: zu ändernde Quellebezeichnung
neue Quellebezeichnung

– db_change_file_zugriff_zuordnung_mID:

Diese Routine ermöglicht die Änderung des Zugriffsrechts, das der übergebenen Datei zugewiesen ist. Der Benutzer wird daraufhin aufgefordert, das zu ändernde Zugriffsrecht und das neue Zugriffsrecht einzugeben, wobei das erstgenannte optional ist. In Abhängigkeit von den Eingaben wird wie folgt unterschieden:

- * zu änderndes Zugriffsrecht nicht eingegeben:
In diesem Fall werden alle Datensätze in der Tabelle FILE_ZUG_ZUORDNUNG geändert, bei denen die übergebene Sprachsignaldatei vorkommt, ungeachtet des partizipierenden Zugriffsrechts. Vor der Änderung muß aber die Bestätigung einer Rückfrage erfolgen.
 - * zu änderndes Zugriffsrecht eingegeben:
In diesem Fall werden lediglich Datensätze aus der Tabelle FILE_ZUG_ZUORDNUNG geändert, die sowohl die übergebene Sprachsignaldatei als auch das zu ändernde Zugriffsrecht enthalten.
- Eingabe: zu änderndes Zugriffsrecht
neues Zugriffsrecht

Option A=5B=7C=1

Zusammenfassung:

Eingabe: Dateiname (obligatorisch)

weitere Eingabemöglichkeiten bei den oben genannten Routinen

Im Gegensatz zu allen anderen Menüpunkten ist es hierbei nicht möglich, die bei diesem Menüpunkt aktivierte Routine im Automatisierungsmodus zu starten. Der Grund dafür besteht in der Ablauflogik der Routine db_change_file_zuordnung. Es werden nämlich der Reihe nach alle potentiellen Zuordnungen für die eingangs angegebene Sprachsignaldatei berücksichtigt und gegebenenfalls modifiziert. Der Benutzer soll also unterschiedliche Argumente angeben, die sich nicht so leicht komplett in eine Befehlskette (wie im Automatisierungsmodus vorgesehen) packen läßt.

• **2: Datei-Format-Zuordnung-Änderungen**

Dieser Menüpunkt startet die Routine db_change_file_format_zuordnung, die genau wie die oben beschriebene db_change_file_format_zuordnung_mID fungiert. D.h., mit diesem Menüpunkt können Änderungen an den Datensätzen der Tabelle FILE_FO_ZUORDNUNG vorgenommen werden. Dementsprechend kann dieser Menüpunkt als ein Teil des Menüpunkts "Dateizuordnungen-Änderungen-Allgemein" aufgefaßt werden. Auf jeden Fall muß der Benutzer einen Dateinamen eingeben. Es folgen Angaben über die zu ändernde Formatkennzeichnung und die neue Formatkennzeichnung, wobei die erstgenannte optional ist. Es werden auch zwei Fälle unterschieden:

- zu ändernde Formatkennzeichnung nicht eingegeben:
In diesem Fall werden alle Datensätze in der Tabelle FILE_FO_ZUORDNUNG geändert, bei denen die übergebene Sprachsignaldatei vorkommt, ungeachtet der partizipierenden Formatkennzeichnung. Vor der Änderung muß aber die Bestätigung einer Rückfrage erfolgen.
- zu ändernde Formatkennzeichnung eingegeben:
In diesem Fall werden lediglich Datensätze aus der Tabelle FILE_FO_ZUORDNUNG geändert, die sowohl die übergebene Sprachsignaldatei als auch die zu ändernde Formatkennzeichnung enthalten.

Option A=5B=7C=2

Zusammenfassung:

Eingabe: Dateiname (obligatorisch)

zu ändernde Formatkennzeichnung

neue Formatkennzeichnung

Automatisierungsmodus:

```
korpo -s "Datenbankname" A=5 B=7 C=2 "Dateinamen"
"alte Formatkennzeichnung" "neue Formatkennzeichnung" -e
oder
korpo -s "Datenbankname" A=5 B=7 C=2 "Dateinamen"
"neue Formatkennzeichnung" -e
```

• 3: Datei-Prozedur-Zuordnung-Änderungen

Dieser Menüpunkt startet die Routine `db_change_file_prozedur_zuordnung`, die genau wie die oben beschriebene `db_change_file_prozedur_zuordnung_mID` fungiert. D.h., mit diesem Menüpunkt können Änderungen an den Datensätzen der Tabelle `FILE_PRO_ZUORDNUNG` vorgenommen werden. Dementsprechend kann dieser Menüpunkt als ein Teil des Menüpunkts "Dateizuordnungen-Änderungen-Allgemein" aufgefaßt werden. Auf jeden Fall muß der Benutzer einen Dateinamen eingeben. Es folgen Angaben über den zu ändernden Prozedurnamen und den neuen Prozedurnamen, wobei der erstgenannte optional ist. Es werden auch zwei Fälle unterschieden:

- zu ändernder Prozedurname nicht eingegeben:
In diesem Fall werden alle Datensätze in der Tabelle `FILE_PRO_ZUORDNUNG` geändert, bei denen die übergebene Sprachsignaldatei vorkommt, ungeachtet des partizipierenden Prozedurnamens. Vor der Änderung muß aber die Bestätigung einer Rückfrage erfolgen.
- zu ändernder Prozedurname eingegeben:
In diesem Fall werden lediglich Datensätze aus der Tabelle `FILE_PRO_ZUORDNUNG` geändert, die sowohl die übergebene Sprachsignaldatei als auch den zu ändernden Prozedurnamen enthalten.

Option A=5B=7C=3

Zusammenfassung:

Eingabe: Dateiname (obligatorisch)
zu ändernder Prozedurname
neuer Prozedurname

Automatisierungsmodus:

```
korpo -s "Datenbankname" A=5 B=7 C=3 "Dateinamen"
"alten Prozedurnamen" "neuen Prozedurnamen" -e
oder
korpo -s "Datenbankname" A=5 B=7 C=3 "Dateinamen"
"neuen Prozedurnamen" -e
```

• 4: Datei-Haupttyp-Zuordnung-Änderungen

Dieser Menüpunkt startet die Routine `db_change_file_haupttyp_zuordnung`, die genau wie die oben beschriebene `db_change_file_haupttyp_zuordnung_mID` fungiert. D.h., mit diesem Menüpunkt können Änderungen an den Datensätzen der Tabelle `FILE_HT_ZUORDNUNG` vorgenommen werden. Dementsprechend kann dieser Menüpunkt als ein Teil des Menüpunkts "Dateizuordnungen-Änderungen-Allgemein" aufgefaßt werden. Auf jeden Fall muß der Benutzer einen Dateinamen eingeben. Es folgen Angaben über die zu ändernde Haupttypkennzeichnung und die neue Haupttypkennzeichnung, wobei die erstgenannte optional ist. Es werden auch zwei Fälle unterschieden:

- zu ändernde Haupttypkennzeichnung nicht eingegeben:
In diesem Fall werden alle Datensätze in der Tabelle FILE_HT_ZUORDNUNG geändert, bei denen die übergebene Sprachsignaldatei vorkommt, ungeachtet des partizipierenden Haupttyps. Vor der Änderung muß aber die Bestätigung einer Rückfrage erfolgen.
- zu ändernde Haupttypkennzeichnung eingegeben:
In diesem Fall werden lediglich Datensätze aus der Tabelle FILE_HT_ZUORDNUNG geändert, die sowohl die übergebene Sprachsignaldatei als auch die zu ändernde Haupttypkennzeichnung enthalten.

Option A=5B=7C=4

Zusammenfassung:

Eingabe: Dateiname (obligatorisch)
zu ändernde Haupttypkennzeichnung
neue Haupttypkennzeichnung

Automatisierungsmodus:

korpo -s "Datenbankname" A=5 B=7 C=4 "Dateinamen"
"alte Haupttypkennzeichnung" "neue Haupttypkennzeichnung" -e
oder
korpo -s "Datenbankname" A=5 B=7 C=4 "Dateinamen"
"neue Haupttypkennzeichnung" -e

• **5: Datei–Untertyp–Zuordnung–Änderungen**

Dieser Menüpunkt startet die Routine db.change_file_untertyp_zuordnung, die genau wie die oben beschriebene db.change_file_untertyp_zuordnung_mID fungiert. D.h., mit diesem Menüpunkt können Änderungen an den Datensätzen der Tabelle FILE_UT_ZUORDNUNG vorgenommen werden. Dementsprechend kann dieser Menüpunkt als ein Teil des Menüpunkts "Dateizuordnungen–Änderungen–Allgemein" aufgefaßt werden. Auf jeden Fall muß der Benutzer einen Dateinamen eingeben. Es folgen Angaben über die zu ändernde Untertypkennzeichnung und die neue Untertypkennzeichnung, wobei die erstgenannte optional ist. Es werden auch zwei Fälle unterschieden:

- zu ändernde Untertypkennzeichnung nicht eingegeben:
In diesem Fall werden alle Datensätze in der Tabelle FILE_UT_ZUORDNUNG geändert, bei denen die übergebene Sprachsignaldatei vorkommt, ungeachtet des partizipierenden Untertyps. Vor der Änderung muß aber die Bestätigung einer Rückfrage erfolgen.
- zu ändernde Untertypkennzeichnung eingegeben:
In diesem Fall werden lediglich Datensätze aus der Tabelle FILE_UT_ZUORDNUNG geändert, die sowohl die übergebene Sprachsignaldatei als auch die zu ändernde Untertypkennzeichnung enthalten.

Option A=5B=7C=5

Zusammenfassung:

Eingabe: Dateiname (obligatorisch)
zu ändernde Untertypkennzeichnung
neue Untertypkennzeichnung

Automatisierungsmodus:

```
korpo -s "Datenbankname" A=5 B=7 C=5 "Dateinamen"
"alte Untertypkennzeichnung" "neue Untertypkennzeichnung" -e
oder
korpo -s "Datenbankname" A=5 B=7 C=5 "Dateinamen"
"neue Untertypkennzeichnung" -e
```

• 6: Datei-Quelle-Zuordnung-Änderungen

Dieser Menüpunkt startet die Routine `db_change_file_quelle_zuordnung`, die genau wie die oben beschriebene `db_change_file_quelle_zuordnung_mID` fungiert. D.h., mit diesem Menüpunkt können Änderungen an den Datensätzen der Tabelle `FILE_QU_ZUORDNUNG` vorgenommen werden. Dementsprechend kann dieser Menüpunkt als ein Teil des Menüpunkts "Dateizuordnungen-Änderungen-Allgemein" aufgefaßt werden. Auf jeden Fall muß der Benutzer einen Dateinamen eingeben. Es folgen Angaben über die zu ändernde Quellebezeichnung und die neue Quellebezeichnung, wobei die erstgenannte optional ist. Es werden auch zwei Fälle unterschieden:

- zu ändernde Quellebezeichnung nicht eingegeben:
In diesem Fall werden alle Datensätze in der Tabelle `FILE_QU_ZUORDNUNG` geändert, bei denen die übergebene Sprachsignaldatei vorkommt, ungeachtet der partizipierenden Quelle. Vor der Änderung muß aber die Bestätigung einer Rückfrage erfolgen.
- zu ändernde Quellebezeichnung eingegeben:
In diesem Fall werden lediglich Datensätze aus der Tabelle `FILE_QU_ZUORDNUNG` geändert, die sowohl die übergebene Sprachsignaldatei als auch die zu ändernde Quellebezeichnung enthalten.

Option A=5B=7C=6

Zusammenfassung:

Eingabe: Dateiname (obligatorisch)
zu ändernde Quellebezeichnung
neue Quellebezeichnung

Automatisierungsmodus:

```
korpo -s "Datenbankname" A=5 B=7 C=6 "Dateinamen"
"alte Quellebezeichnung" "neue Quellebezeichnung" -e
oder
korpo -s "Datenbankname" A=5 B=7 C=6 "Dateinamen"
"neue Quellebezeichnung" -e
```

• 7: Datei-Zugriff-Zuordnung-Änderungen

Dieser Menüpunkt startet die Routine `db_change_file_zugriff_zuordnung`, die genau wie die oben beschriebene `db_change_file_zugriff_zuordnung_mID` fungiert. D.h., mit diesem Menüpunkt können Änderungen an den Datensätzen der Tabelle `FILE_ZUG_ZUORDNUNG` vorgenommen werden. Dementsprechend kann dieser Menüpunkt als ein Teil des Menüpunkts "Dateizuordnungen-Änderungen-Allgemein" aufgefaßt werden. Auf jeden Fall muß der Benutzer einen Dateinamen eingeben. Es folgen Angaben über die zu ändernde Zugriffsrecht-ID und die neue Zugriffsrecht-ID, wobei die erstgenannte optional ist. Es werden auch zwei Fälle unterschieden:

- zu ändernde Zugriffsrecht-ID nicht eingegeben:
In diesem Fall werden alle Datensätze in der Tabelle FILE_ZUG_ZUORDNUNG geändert, bei denen die übergebene Sprachsignaldatei vorkommt, ungeachtet des zugewiesenen Zugriffsrechts. Vor der Änderung muß aber die Bestätigung einer Rückfrage erfolgen.
- zu ändernde Zugriffsrecht-ID eingegeben:
In diesem Fall werden lediglich Datensätze aus der Tabelle FILE_ZUG_ZUORDNUNG geändert, die sowohl die übergebene Sprachsignaldatei als auch die zu ändernde Zugriffsrecht-ID enthalten.

Option A=5B=7C=7

Zusammenfassung:

Eingabe: Dateiname (obligatorisch)
zu ändernde Zugriffsrecht-ID
neue Zugriffsrecht-ID

Automatisierungsmodus:

```
korpo -s "Datenbankname" A=5 B=7 C=6 "Dateinamen"  
"alte Zugriffsrecht-ID" "neue Zugriffsrecht-ID" -e  
oder  
korpo -s "Datenbankname" A=5 B=7 C=6 "Dateinamen"  
"neue Zugriffsrecht-ID" -e
```

3.4.19 Menü A5B13_C

Wie bereits erwähnt ist dieses Menüsystem verantwortlich für Änderungen in der Tabelle FILEEXT_PROZEDUR_ZU. Es existieren zwei Asugangspunkte, von wo Änderungen vorgenommen werden können: ausgehend von einem Prozedurnamen und ausgehend von einer Dateiextension.

- **1: Prozedur-Dateiextension-Zuordnung-Prozedurnamen**

Dieser Menüpunkt bedient sich der Routine db_change_filext_prozedur_zu_pro und führt Änderungen an Prozedurnamen in der Tabelle FILEEXT_PROZEDUR_ZU aus. Zunächst wird der zu ändernde Prozedurname erwartet. Dann folgt die Angabe über den neuen Prozedurnamen. Es muß hier darauf hingewiesen werden, daß diese zwei Datenelemente erforderlich sind. Danach kommt die Aufforderung, eine Dateiextension einzugeben. Im Gegensatz zu den Prozedurnamen ist die Dateiextension nur optional. Anschließend wird wie folgt vorgegangen:

- Dateiextension nicht eingegeben In diesem Fall werden alle Datensätze geändert, die nur den zu ändernden Prozedurnamen besitzen, unabhängig von deren zugewiesenen Dateiextensionen. Auf diese Weise kann auf einen Schlag eingestellt werden, welche Prozedur mit welchen Dateiextensionen operiert kann. Es müs wie üblich die Bestätigung einer Sicherheitsrückfrage vor der Änderung erfolgen.
- Dateiextension eingegeben In diesem Fall hat man die Möglichkeit, einzustellen, welche Prozedur auf die eingegebene Dateiextension angewandt werden soll. Dies bedeutet konkret, daß nur diejenige Datensätze geändert werden können, die sowohl den zu ändernden Prozedurnamen als auch die angegebene Dateiextension haben.

Option A=5B=13C=1

Zusammenfassung:

Eingabe: zu ändernder Prozedurname (obligatorisch)
 neuer Prozedurname (obligatorisch)
 Dateiextension (optional)

Automatisierungsmodus:

```
korpo -s "Datenbankname" A=5 B=13 C=1 "Prozedurnamen"
"neuen Prozedurnamen" "Dateiextension" -e
oder
korpo -s "Datenbankname" A=5 B=13 C=1 "Prozedurnamen"
"neuen Prozedurnamen" -e
```

- **2: Prozedur-Dateiextension-Zuordnung-Dateiextension**

Dieser Menüpunkt bedient sich der Routine `db_change_filext_prozedur_zu_ext` und führt Änderungen an Dateiextensionen in der Tabelle `FILEEXT_PROZEDUR_ZU` aus. Zunächst wird die zu ändernde Dateiextension erwartet. Dann folgt die Angabe über die neue Dateiextension. Es muß hier darauf hingewiesen werden, daß diese zwei Datenelemente erforderlich sind. Danach kommt die Aufforderung, einen Prozedurnamen einzugeben. Im Gegensatz zu den Dateiextensionen ist der Prozedurname nur optional. Anschließend wird wie folgt vorgegangen:

- Prozedurname nicht eingegeben:
 In diesem Fall werden alle Datensätze geändert, die nur die zu ändernde Dateiextension besitzen, unabhängig von deren assoziierten Prozedurnamen. Auf diese Weise kann auf einmal eingestellt werden, welche Dateiextension mit welchen Prozedurnamen assoziiert. Es müs wie üblich die Bestätigung einer Sicherheitsrückfrage vor der Änderung erfolgen.
- Prozedurname eingegeben:
 In diesem Fall hat man die Möglichkeit, einzustellen, welche Dateiextension mit dem eingegebenen Prozedurnamen assoziieren soll. Dies bedeutet konkret, daß nur diejenige Datensätze geändert werden können, die sowohl die zu ändernde Dateiextension als auch den angegebenen Prozedurnamen haben.

Option A=5B=13C=2

Zusammenfassung:

Eingabe: zu ändernde Dateiextension (obligatorisch)
 neue Dateiextension (obligatorisch)
 Prozedurname (optional)

Automatisierungsmodus:

```
korpo -s "Datenbankname" A=5 B=13 C=2 "Dateiextension"
"neue Dateiextension" "Prozedurnamen" -e
oder
korpo -s "Datenbankname" A=5 B=13 C=2 "Dateiextension"
"neue Dateiextension" -e
```

- **3: Prozedur-Dateiextension-Zuordnung-Dateiextension-Ziel**

Dieser Menüpunkt bedient sich der Routine `db_change_filext_prozedur_zu_zielext` und führt Änderungen an Zieldateiextensionen in der Tabelle `FILEEXT_PROZEDUR_ZU`

aus. Zunächst wird die zu ändernde Zieldateiextension erwartet. Dann folgt die Angabe über die neue Zieldateiextension. Es muß hier darauf hingewiesen werden, daß diese zwei Datenelemente erforderlich sind. Danach kommt die Aufforderung, einen Prozedurnamen einzugeben. Im Gegensatz zu den Zieldateiextensionen ist der Prozedurname nur optional. Anschließend wird wie folgt vorgegangen:

- Prozedurname nicht eingegeben:
In diesem Fall werden alle Datensätze geändert, die nur die zu ändernde Dateiextension besitzen, unabhängig von deren assoziierten Prozedurnamen. Auf diese Weise kann auf einmal eingestellt werden, welche Zieldateiextension mit welchen Prozedurnamen assoziiert. Es müs wie üblich die Bestätigung einer Sicherheitsrückfrage vor der Änderung erfolgen.
- Prozedurname eingegeben:
In diesem Fall hat man die Möglichkeit, einzustellen, welche Zieldateiextension mit dem eingegebenen Prozedurnamen assoziieren soll. Dies bedeutet konkret, daß nur diejenige Datensätze geändert werden können, die sowohl die zu ändernde Dateiextension als auch den angegebenen Prozedurnamen haben.

Option A=5B=13C=3

Zusammenfassung:

Eingabe: zu ändernde Zieldateiextension (obligatorisch)
neue Zieldateiextension (obligatorisch)
Prozedurname (optional)

Automatisierungsmodus:

```
korpo -s "Datenbankname" A=5 B=13 C=3 "Zieldateiextension"
"neue Zieldateiextension" "Prozedurnamen" -e
oder
korpo -s "Datenbankname" A=5 B=13 C=3 "Zieldateiextension"
"neue Zieldateiextension" -e
```

3.4.20 Menü A1B1C12_D

- **1: Dateianzahl - Allgemein**

Mit Hilfe dieses Menüpunkts kann die Gesamtanzahl der in der Tabelle FILE enthaltenen Sprachsignaldateien ermittelt und angezeigt werden. Das Ergebnis wird in einer Outputdatei abgespeichert. Voreingestellt heißt die Outputdatei "erg.dat", welche auch im Automatisierungsmodus verwendet wird.

Option A=1B=1C=12D=1

Zusammenfassung:

Eingabe: keine

Automatisierungsmodus:

```
korpo -s "Datenbankname" A=1 B=1 C=12 D=1 -e
```

- **2: Dateianzahl – Stamm**

Mit Hilfe dieses Menüpunkts kann die Gesamtanzahl der in der Tabelle FILE enthaltenen Sprachsignaldateien ermittelt und angezeigt werden, welche den angegebenen

Stamm besitzen. Das Ergebnis wird in einer Outputdatei abgespeichert. Voreingestellt heißt die Outputdatei "erg.dat", welche auch im Automatisierungsmodus verwendet wird.

Option A=1B=1C=12D=2

Zusammenfassung:

Eingabe: Dateistamm (obligatorisch)

Automatisierungsmodus:

korpo -s "Datenbankname" A=1 B=1 C=12 D=2 "Dateistamm" -e

• **3: Dateianzahl – Datum**

Mit Hilfe dieses Menüpunkts kann die Gesamtanzahl der in der Tabelle FILE enthaltenen Sprachsignaldateien ermittelt und angezeigt werden, welche das angegebene Datum haben. Das Ergebnis wird in einer Outputdatei abgespeichert. Voreingestellt heißt die Outputdatei "erg.dat", welche auch im Automatisierungsmodus verwendet wird.

Option A=1B=1C=12D=3

Zusammenfassung:

Eingabe: Datum (obligatorisch)

Automatisierungsmodus:

korpo -s "Datenbankname" A=1 B=1 C=12 D=3 "Datum" -e

• **4: Dateianzahl – Uhrzeit**

Mit Hilfe dieses Menüpunkts kann die Gesamtanzahl der in der Tabelle FILE enthaltenen Sprachsignaldateien ermittelt und angezeigt werden, welche die angegebene Uhrzeit haben. Das Ergebnis wird in einer Outputdatei abgespeichert. Voreingestellt heißt die Outputdatei "erg.dat", welche auch im Automatisierungsmodus verwendet wird.

Option A=1B=1C=12D=4

Zusammenfassung:

Eingabe: Uhrzeit (obligatorisch)

Automatisierungsmodus:

korpo -s "Datenbankname" A=1 B=1 C=12 D=4 "Uhrzeit" -e

• **5: Dateianzahl – Endung**

Mit Hilfe dieses Menüpunkts kann die Gesamtanzahl der in der Tabelle FILE enthaltenen Sprachsignaldateien ermittelt und angezeigt werden, welche die angegebene Dateierweiterung haben. Das Ergebnis wird in einer Outputdatei abgespeichert. Voreingestellt heißt die Outputdatei "erg.dat", welche auch im Automatisierungsmodus verwendet wird.

Option A=1B=1C=12D=5

Zusammenfassung:

Eingabe: Dateiextension (obligatorisch)

Automatisierungsmodus:

korpo -s "Datenbankname" A=1 B=1 C=12 D=5 "Dateiextension" -e

• 6: Dateianzahl – Quelle

Mit Hilfe dieses Menüpunkts kann die Gesamtanzahl der in der Tabelle FILE enthaltenen Sprachsignaldateien ermittelt und angezeigt werden, welche von der angegebenen Quelle stammen. Das Ergebnis wird in einer Outputdatei abgespeichert. Voreingestellt heißt die Outputdatei "erg.dat", welche auch im Automatisierungsmodus verwendet wird.

Option A=1B=1C=12D=6

Zusammenfassung:

Eingabe:Quelle (obligatorisch)

Automatisierungsmodus:

korpo -s "Datenbankname" A=1 B=1 C=12 D=6 "Quelle" -e

Kapitel 4

Perl–Schnittstelle

Das Perl–Programm ist ein eigenständiges Programm, bildet eine Verbindungsschnittstelle mit dem anderen C–Programm, indem das Perl–Programm Inputdaten von dem anderen geliefert bekommt und diese Daten weiter bearbeitet. Dabei bestehen die benötigten Inputdaten von den Abfragergebnissen, die durch eine gezielte Anfrage vom C–Programm zurückgeliefert werden. Infolgedessen müssen Inputdaten für die Perl–Schnittstelle vorher bereitgestellt werden.

4.1 Programmstart mit Kommandozeileoptionen

Das Perl–Programm ist mit Absicht so gestaltet, daß ihm eine Reihe von Kommandozeileoptionen übergeben wird. **Der Programmstart erfolgt mit der Angabe des Programmnamens "analyse", gefolgt von Kommandozeileoptionen und der Datei, in der gesucht wird.** Es wird dabei stillschweigend vorausgesetzt, daß das Perl–Programm ausführbares Zugriffsrecht zugewiesen bekommt (etwa durch `chmod +x analyse.pl`). Als Alternative dazu kann man dem Namen des Perl–Programms der Aufruf des Perl–Intepreters voranstellen (etwa `perl analyse.pl ...`). Darüber hinaus muß natürlich Perl 5 auf dem Rechner installiert sein.

Folgende Liste beschreibt die in Frage kommenden Kommandozeileoptionen:

- i** Mit der Kommandozeileoption "-i" wird die eingebaute Hilfe aktiviert. Der Benutzer erhält eine Liste der möglichen Kommandozeileoptionen angezeigt.
- r** Ergebnisdatei Die Kommandozeileoption "-r" gibt die Outputdatei an, die das Ergebnis nach Beenden des Programm enthält. Wenn diese Kommandozeileoption fehlt, wird automatisch die Outputdatei namens *out.dat* herangezogen. Dies ist die Voreinstellung.
- v** Dies ist der Schalter für Debug–Informationen. D.h., nützliche Informationen werden damit zur Laufzeit des Programms angezeigt.
- k** Definitionsdatei Mit dieser Kommandozeileoption wird die Definitionsdatei für Features (z.B. Vokal, Frikative, stimmlose, stimmhafte Lauten) und assoziierte Elemente (Segmente, Laute, Sylben) angegeben. Diese Datei ist für die Analyse von Sprachsignalen notwendig, bei der nach bestimmten linguistischen Merkmalen gesucht wird (Sprachlaute beispielsweise). Fehlt diese Angabe, wird als Voreinstellung die Suche nach Wörtern aktiviert. Dabei ist es zu beachten, daß die Definitionsdatei in demselben Verzeichnis zu finden ist, wo sich das Perl–Analyseprogramm befindet. Die Angabe [-k] und [-F] müssen immer **paarweise** gemacht werden. In folgenden Abschnitten wird darüber mehr erläutert.

-pro "Format-String" Mit Hilfe dieser Kommandozeileoption kann eine Zeichenkette übergeben werden, die in der Outputdatei mit gespeichert wird. Alles, was diese Zeichenkette spezifiziert, wird wörtlich und genau in derselben Reihenfolge der Elemente innerhalb der Zeichenkette in die Outputdatei übernommen. Auf diese Weise kann man die Kommandozeile samt Argumenten für ein beliebiges startbares Programm in eine Zeichenkette packen, das mit den gespeicherten Daten später aufgerufen werden kann (flexible Formatierung).

Die Zeichenkette kann wiederum folgende Optionen enthalten, die ihrerseits optional sind:

- Namen des Programms
- %S : Startzeit
- %E : Endzeit
- %N : Startzeit des nächstfolgenden Sprachsegments
- %-d : Startzeit vom vorhergehenden Sprachsegment im Abstand d vom aktuellen Abtastwert (d ist eine Interger-Zahl). Das Minus-Zeichen muß dabei unbedingt eingegeben werden
- %d oder \$+d : Startzeit vom nächstfolgenden Sprachsegment im Abstand d vom aktuellen Abtastwert. Das Plus-Zeichen ist in diesem Fall aber optional
- %I : das assoziierte Sprachsegment, das gefunden wird
- %F : Name der Datei, in der das Sprachsegment gefunden wird
- %% : stellvertretend für ein %-Zeichen

Fehlt die Angabe überhaupt, wird das Programm *sgplay* mitsamt einigen voreingestellten Argumenten als Kommandozeile in der Outputdatei gespeichert. Das voreingestellte Format setzt sich also zusammen aus folgenden Bestandteilen:

- sgplay
- -s %S
- -e %E
- %F
- %I

-F "Feature-String" Diese Kommandozeileoption soll bei der Suche nach Features angegeben werden. Features werden in einem Feature-String zusammengefaßt. Dabei können die Features miteinander durch Operatoren verknüpft sein. Es existieren zwei Operatoren:

- | bedeutet Oder-Verknüpfung. Diese Operation kann als Vereinigungsoperation interpretiert werden. D.h., das Endergebnis wird über alle Teilergebnisse gebildet, enthält also alle Elemente der Teilergebnisse.
- & bedeutet Und-Verknüpfung. Diese Operation kann als Schnittmengeoperation angedeutet werden. Das Endergebnis enthält nur diejenige Elemente, die in allen Teilergebnissen vorkommen.

-E "Elemente-String" Diese Kommandozeileoption soll bei der Suche nach Elementen angegeben werden. Dabei sind Elemente beispielsweise bestimmte Sprachlauten.

Im Gegensatz zur Kommandozeileoption [-F] wird bei der Suche nach Sprachlementen (Wörtern) nur der Operator — unterstützt, denn es ist offensichtlich nicht sinnvoll, nach der Schnittmenge von Sprachlementen zu suchen. Diese Option kann allein oder zusammen mit der Option [-F] angegeben werden. Falls nur [-E] vorliegt, wird die Suche nach Wörtern (ohne Definitionsdatei, also ohne Angabe von [-k]) aktiviert. Im anderen Fall wird die kombinierte Suche nach Wörtern und Features gestartet.

Entsprechende Fehlermeldungen werden im Normalfall angezeigt, wenn Kommandozeileoptionen nicht definitionskonform oder mit falscher Syntax eingegeben werden (wie z.B. bei Eingabe von `analyse.pl` ohne Kommandozeileoptionen). Die Priorität der zwei Operatoren `|` und `&` wird durch das Setzen von Klammerpaaren untereinander geregelt. Die Abarbeitung der Feature- und Elemente-Angabe erfolgt in der Regel von links nach rechts. Dabei müssen die Feature- und Elemente-Strings in Anführungszeichen gesetzt werden, wodurch der Befehlsinterpreter-Mechanismus von UNIX nicht greift. Überdies müssen die Feature- und Elemente-Strings von den jeweiligen Optionen (`-F`, `-E`) durch Leerzeichen abgetrennt werden. Es ist an dieser Stelle anzumerken, daß die Reihenfolge der durch die Kommandozeileoption `[-pro]` spezifizierten Befehle unverändert bleibt bei der Ausgabe in die Outputdatei und die Suche nach Elementen und Features **nicht case-sensitiv** ist. Darüber hinaus muß zumindest eine der Optionen von `[-F]` und `[-E]` angegeben sein (siehe auch 4.7.1. Die Kommandozeileoptionen). Im Sinne, daß ein Wertebereich spezifiziert werden kann, der das gefundene Sprachsegment umgibt (relative Segmentierung). Die Option `[-p]` hat eine besondere Bedeutung, entspricht dem aktuellen gefundenen Sprachsegment.

4.2 Funktionalität

Es ist nicht genug zu betonen, daß das Perl-Programm der Analyse von Sprachsignalen dient, welche als Abfrageergebnisse vom C-Programm stammen. Die Analyse schließt sich an die Ausführung des C-Programms an. In der Regel sind die Abfrageergebnisse eine Liste, in der Label- oder Phones-Dateien aufgelistet sind. Innerhalb dieser Sprachsignaldateien wird im Zusammenhang mit den übergebenen Kommandozeileoptionen nach Wörtern oder Features und/oder Elementen gesucht. Das Suchergebnis wird wiederum in die Outputdatei geschrieben.

4.3 Aufbau der Definitionsdatei

Die Definitionsdatei ist erforderlich für die Suche nach Features und/oder Elementen und wird eingeleitet durch Angabe der Kommandozeileoption `[-k Definitionsdatei]`. Diese Datei wird zum Beispiel wie folgt definiert:

```
# vo,fr,sth,stl,na
a vo
c fr
z fr
e vo, sth
f fr , stl
i: vo
t stl
n na
p stl
es wort
```

Ein Kommentar wird gekennzeichnet durch ein vorangestelltes Nummerzeichen `#`. Jede Zeile beginnt mit einem Sprachelement wie z.B. `"a"`, `"c"`, `"i:"` u.s.w, gefolgt von den damit verbundenen Features. Beispielsweise ist das Sprachelement `"a"` ein Vokal, `"e"` auch ein Vokal und zugleich stimmhaft. Als Trennzeichen zwischen den Features können Tabulator-, Leerzeichen oder Komma sein. Allerdings kommen nur Leer- oder Tabulator-Zeichen als Trennzeichen zwischen einem Sprachelement und dessen Features. Die Definition jedes Sprachelements beginnt immer mit einer neuen Zeile in der

Definitionsdatei, die Anzahl der Zeilen wird nach oben nicht begrenzt.

Die Definitionsdatei kann auch benutzt werden, um nach Wörtern zu suchen (nicht nur Sprachlaute wie a, i: ...). In diesem Fall wird ein Wort als ein normales Sprachelement spezifiziert und dessen Feature muß ebenfalls entsprechend gekennzeichnet werden. Beispielsweise enthält die obige Definitionsdatei die Spezifikation des Wortes "es" nebst dessen Feature "wort" (beispielsweise). Somit kann man mit Hilfe der Definitionsdatei auf flexible Weise sowohl nach Sprachlauten als auch nach Wörtern suchen.

Im Perl-Programm wird der Inhalt der Definitionsdatei eingelesen und in einer geeigneten Hash-Struktur untergebracht, die zum internen programmtechnischen Zweck erzeugt wird (u.a. zum Vergleich von Features-Mustern).

4.4 Aufbau der Inputdatei

Wie bereits erwähnt enthält die Inputdatei Abfrageergebnisse vom C-Programm. In der Regel sind die betroffenen Abfragen solche, bei denen nach Label- oder Phones- oder Words-Dateien in der Tabelle FILE gesucht wird, die bestimmten Kriterien genügen.

Words-Dateien sind benötigt für die Suche nach bestimmten Wörtern (bei Angabe der Kommandooption [-E] und beim gleichzeitigen Fehlen der Kommandozeileoption [-k])

Label- oder Phones-Dateien werden für die Suche nach Features und Sprachelementen herangezogen (bei Angabe der Kommandozeileoption [-k] und [-F] und/oder [-E])

In der Inputdatei werden die Words- oder Label- oder Phones-Dateien *zeilensweise* aufgelistet.

Folgendes Beispiel zeigt eine Inputdatei, die Phones-Dateien enthält:

```
#Datei für generalisierte Abfrage
/graugans/cd/stern1/phones/df960724.1657.phones
/graugans/cd/stern1/phones/df960731.1657.phones
/graugans/cd/stern1/phones/df960801.1656.phones
/graugans/cd/stern1/phones/df960802.1656.phones
/graugans/cd/stern1/phones/df960803.1656.phones
/graugans/cd/stern1/phones/df960804.1656.phones
/graugans/cd/stern1/phones/df960805.1656.phones
```

Das zweite Beispiel zeigt den Inhalt einer Inputdatei, die Words-Dateien beinhaltet:

```
/graugans/cd/Sternzeit3/April_97/df970430.1656.words
/graugans/cd/Sternzeit3/April_97/df970429.1656.words
/graugans/cd/Sternzeit3/April_97/df970426.1656.words
/graugans/cd/Sternzeit3/April_97/df970425.1656.words
/graugans/cd/Sternzeit3/April_97/df970424.1656.words
/graugans/cd/Sternzeit3/April_97/df970423.1656.words
/graugans/cd/Sternzeit3/April_97/df970422.1656.words
/graugans/cd/Sternzeit3/April_97/df970421.1656.words
/graugans/cd/Sternzeit3/April_97/df970420.1656.words
/graugans/cd/Sternzeit3/April_97/df970419.1656.words
/graugans/cd/Sternzeit3/April_97/df970418.1656.words
```

```
/graugans/cd/Sternzeit3/April_97/df970417.1656.words
/graugans/cd/Sternzeit3/April_97/df970416.1656.words
/graugans/cd/Sternzeit3/April_97/df970415.1656.words
```

4.5 Aufbau der Outputdatei

Im Prinzip enthält die Outputdatei das Suchergebnis und orientiert sich in ihrer Gestaltung nach der Kommandozeileoption [-pro]. Fehlt diese Kommandozeileoption, wird automatisch der festgelegte Kommandostring "sgplay -s Startzeit -e Endzeit Filename Element" eingesetzt. Andernfalls wird der von der Kommandozeileoption [-pro] spezifizierte String herangezogen.

4.6 Beispiele

In diesem Abschnitt werden im Vorgriff auf die Testphase einige Beispiele gezeigt, wie das Perl-Programm bei der Analyse richtig eingesetzt wird.

4.6.1 Features-basierte Suche

```
analyse -v -F "vo|sth" -k defi.dat -r out1.dat erg.dat
```

Die gesuchte Datei ist erg.dat. Es wird nach den Sprachelementen gesucht, die sowohl Vokal als auch Frikativ sind. Die Definitionsdatei ist defi.dat, die Outputdatei heißt out1.dat, Debuginformationen werden angezeigt.

Im folgenden wird ein Teil der Outputdatei out1.dat aufgelistet, in dem ersichtlich wird, daß ohne Kommandozeileoption [-pro] die Voreinstellung "sgplay -s Startzeit -e Endzeit Filename # Element" gilt.

```
sgplay -s 127.380000 -e 127.430000 dlf.sd # e
sgplay -s 127.770000 -e 128.020000 dlf.sd # a
sgplay -s 128.650000 -e 128.720000 dlf.sd # a
sgplay -s 132.660000 -e 132.740000 dlf.sd # e
sgplay -s 132.660000 -e 132.740000 dlf.sd # e:
sgplay -s 132.790000 -e 132.960000 dlf.sd # a
sgplay -s 133.330000 -e 133.360000 dlf.sd # e
```

```
analyse -v -F "vo|sth" -pro "sgplay -s %S -e %E %N %F %I" -k defi.dat -r out1.dat erg.dat
```

Genau wie oben, allerdings kommt in diesem Beispiel die Kommandozeileoption [-pro] ins Spiel. Folgender Auszug der Outputdatei out1.dat zeigt die Auswirkung der Kommandozeileoption [-pro]:

```
sgplay -s 127.380000 -e 127.430000 127.430000 dlf.sd e
sgplay -s 127.770000 -e 128.020000 128.020000 dlf.sd a
sgplay -s 128.650000 -e 128.720000 128.720000 dlf.sd a
sgplay -s 132.660000 -e 132.740000 132.740000 dlf.sd e
sgplay -s 132.660000 -e 132.740000 132.740000 dlf.sd e:
sgplay -s 132.790000 -e 132.960000 132.960000 dlf.sd a
sgplay -s 133.330000 -e 133.360000 133.360000 dlf.sd e
```

4.6.2 Segment–basierte Suche

```
analyse -v -E "es|war" -r out1.dat erg.dat
```

Dieses Beispiel zeigt die Suche nach den zwei Wörtern "es" und "war". Es gilt zu beachten, daß in diesem Fall die Kommandozeileoption [-k] unbedingt fehlen muß.

Das Ergebnis des Programmlaufs sieht wie folgt aus:

```
sgplay -s 137.960000 -e 138.330000 dfl.sd # es
sgplay -s 139.860000 -e 140.370000 dfl.sd # es
sgplay -s 141.720000 -e 141.880000 dfl.sd # es
sgplay -s 142.090000 -e 142.570000 dfl.sd # es
sgplay -s 145.560000 -e 145.740000 dfl.sd # es
sgplay -s 145.740000 -e 146.370000 dfl.sd # es
sgplay -s 155.850000 -e 156.010000 dfl.sd # war
sgplay -s 162.490000 -e 162.880000 dfl.sd # es
sgplay -s 164.840000 -e 165.060000 dfl.sd # es
sgplay -s 167.180000 -e 167.380000 dfl.sd # war
```

```
analyse -v -E "es|war" -pro %sgplay -s %S -e %E %1r out1.dat erg.dat
```

Dieses Beispiel zeigt ebenfalls die Suche nach den zwei Wörtern "es" und "war" wie das vorangehende, allerdings mit dem Unterschied, daß hierbei die Kommandozeileoption [-pro] aktiviert ist.

Der folgende Auszug der Outputdatei zeigt damit auch den Unterschied:

```
sgplay -s 137.960000 -e 138.330000 138.330000
sgplay -s 139.860000 -e 140.370000 140.370000
sgplay -s 141.720000 -e 141.880000 141.880000
sgplay -s 142.090000 -e 142.570000 142.570000
sgplay -s 145.560000 -e 145.740000 145.740000
sgplay -s 145.740000 -e 146.370000 146.370000
sgplay -s 155.850000 -e 156.010000 156.010000
sgplay -s 162.490000 -e 162.880000 162.880000
sgplay -s 164.840000 -e 165.060000 165.060000
sgplay -s 167.180000 -e 167.380000 167.380000
```

4.6.3 Kombinierte Suche

```
analyse -v -F "vo|sth" -E "f" -k defl.dat -r out1.dat erg.dat
```

In diesem Beispiel wird nach Sprachelementen gesucht, deren Features zur Kategorie Vokal und Stimmhaft gehören, und zusätzlich wird nach dem Sprachelement "f" gesucht.

Der folgende Auszug der Outputdatei zeigt das Ergebnis der Suche:

```
sgplay -s 123.160000 -e 123.290000 dfl.sd # f
sgplay -s 125.000000 -e 125.080000 dfl.sd # f
sgplay -s 125.440000 -e 125.510000 dfl.sd # f
sgplay -s 127.380000 -e 127.430000 dfl.sd # e
sgplay -s 127.770000 -e 128.020000 dfl.sd # a
sgplay -s 128.650000 -e 128.720000 dfl.sd # a
sgplay -s 132.660000 -e 132.740000 dfl.sd # e
sgplay -s 132.660000 -e 132.740000 dfl.sd # e
```

4.7 Wichtige Bemerkungen zur Perl-Schnittstelle

Es sollen in diesem Abschnitt einige Bemerkungen festgehalten werden, die für das Verständnis der Arbeitsweise des Perl-Programms von wichtiger Bedeutung sind.

4.7.1 Allgemeine Bemerkungen

Wie bereits angesprochen fungiert die Perl-Schnittstelle als eine eigene Schnittstelle zum C-Programm, die Inputwerte in Form von C-Programm erstellten Dateilisten entgegennimmt. Dies hat zur Folge, daß Änderungen im Programmablauf und/oder in Funktionalität des C-Programms keine unmittelbaren Änderungen in Perl-Schnittstelle mit sich ziehen, sofern die Inputwerte für die Perl-Schnittstelle unverändert bleiben. Soll in absehbarer Zukunft das Programm erweitert werden, indem eine neue Version von mSQL eingesetzt wird, ist die Perl-Schnittstelle ohne Veränderung verwendbar, gesetzt den Fall, daß die Funktionalität des C-Programms ebenfalls unverändert bleibt.

Aufgrund des inhaltlichen Formats der Sprachsignaldateien, in denen nach bestimmten Features bzw. bestimmten Elementen gesucht und darüber analysiert wird, wird ein Feature oder ein Sprachelement nur als ganzes Wort gesucht (also kein Teilwort). Darüber hinaus gestaltet sich die Suche ohne Case sensitive-Unterscheidung. D.h, Groß- und Kleinbuchstaben werden als gleich betrachtet.

Die Angabe der Operatoren wird ebenfalls kontrolliert. Beispielsweise erfolgt eine Fehlermeldung, sofern der Operator doppelt vorkommt.

Hinsichtlich der Angabe von Features und Elementen mit der Option [-F] bzw. [-E] muß der Benutzer dafür sorgen, daß das Setzen von Klammerpaaren richtig erfolgt, falls die Priorität der Abarbeitung anderweitig interpretiert werden soll.

Bei Analysen unter Mitwirkung einer Definitionsdatei (Option [-k]) bringt die Perl-Schnittstelle entsprechende Meldungen zum Vorschein, falls die gesuchten Features in der Definitionsdatei nicht spezifiziert sind. Wenn die Rede von der Konfigurations- oder Definitionsdatei ist, dann handelt es sich nämlich um dieselbe Datei, welche die Definition der Features enthält.

Bei Angabe der Kommandozeileoption [-pro] soll der Name eines Programms (beispielsweise sgplay) spezifiziert sein. Dies liegt aber im Ermessen des Benutzers und kann nützlich sein, wenn lediglich bestimmte Informationen von den Sprachsignaldateien extrahiert werden sollen. Das Analyse-Programm kann die Kommandozeilenkette, die bei [-pro] angegeben ist, nicht interpretieren. D.h., das Programm betrachtet die Kommandozeilenkette als eine einfache Zeichenkette. Dies bedeutet aber auch, daß es in der Verantwortung des Benutzers liegt, eine sinnvolle Kommandozeilenkette anzugeben.

Alle spezifizierten Platzhalter in der Kommandozeileoption [-pro] werden komplett durch entsprechende Werte ersetzt. Es findet also eine globale Wertersetzung statt.

Es wird versucht, mögliche Eingabefehler in Bezug auf die Angabe der Kommandozeileoptionen abzufangen und gegebenenfalls entsprechende Meldungen anzuzeigen. In diesem Zusammenhang kann die Kommandozeileoption [-i] von wichtiger Bedeutung sein.

4.7.2 Bemerkungen zur Wertersetzung bei Kommandozeileoption %F

Zur Erinnerung steht an der Stelle der Kommandozeile %F in der Outputdatei der vollständige Name einer Sprachsignaldatei (zusammen mit dem Pfad). Überdies wird die Dateierweiterung automatisch nach

”Sd” umbenannt, unabhängig davon, in welchen Dateitypen (Words-, Tags-, Phones-Format ...) die Suche vorgenommen wird. Somit kann das voreingestellte Programm ”sgplay” ohne weiteres gestartet werden. Die verteilte Korpora besteht nämlich aus verstreut vorliegenden Datenbeständen. Aus diesem Grund haben verschiedene Sprachdateien verschiedener Formate auch verschiedene Pfade. Aufgrund der Umbenennung der Dateiextension nach ”Sd” während der Datenauswertung stimmt aber der Pfad nicht mehr und soll aktualisiert werden.

Beispiel:

- Suche in Words-Dateien unter dem Pfad /graugans/cd/stern1/words/
- Ergebnis: sgplay ... /graugans/cd/stern1/words/dlf970324.1656.sd ...
D.h. Die Datei dlf970324.1656.words unter dem Pfad /graugans/cd/stern1/words/ wird nach dlf970324.1656.sd umbenannt. Das Verzeichnis /graugans/cd/stern1/words enthält die Datei dlf970324.1656.sd aber nicht. Diese Datei ist aber unter dem Verzeichnis /graugans/cd/stern1/sd/ zu finden.

Abhilfe leistet das Perl-Programm ”change-path.pl” (siehe auch 6.2.3). Mit Hilfe des Hilfsprogramms ”change-path.pl” kann das Unterverzeichnis ”words nach ”sd” umbenannt werden (/graugans/cd/stern1/words/ wird /graugans/cd/stern1/sd/). Auf diese Weise kann das Programm ”sgplay” ohne Problem sofort gestartet werden. Es muß allerdings vorausgesetzt werden, daß jedes Words-Datei eine entsprechende Sd-Datei hat. Der Aufruf des Hilfsprogramms ”change-path.pl” lautet wie folgt:

```
change-path.pl Inputdatei altes Wort neues Wort
Beispiel: change-path.pl words.dat words sd
```

4.7.3 Bemerkungen zu Text-Dateien

Bei der Analyseauswertung im Fall von Text-Dateien ist eine Differenz feststellbar. Der Grund dafür besteht im andersartigen Aufbau der Text-Dateien und im damit verbundenen Aufbau der Ergebnisdatei (in der Ergebnisdatei wird nämlich die Textzeile angezeigt, in der die gesuchten Wörter liegen). Aufgrund dieser Besonderheit kann der Fall auftreten, daß eine Textzeile bei checkele.pl doppel aufgezählt, sofern die **verschiedenen** gesuchten Wörtern in derselben Textzeile liegen. Beispielsweise kommt diejenige Textzeile bei der Suche nach den Wörtern ”Stern” und ”Mitte” im Ergebnisdatei doppelt vor, da in dieser Textzeile sowohl Stern als auch Mitte vorliegt. Folgender Ausschnitt aus der Ergebnisdatei verdeutlicht dies:

```
Zeile:gleichmäßig einen Stern in ihrer Mitte. Es gibt jedoch: /graugans/cd/stern1/texte/dlf960816.1656.txt # Stern 14
Zeile:gleichmäßig einen Stern in ihrer Mitte. Es gibt jedoch: /graugans/cd/stern1/texte/dlf960816.1656.txt # Mitte 15
```

Unter diesem Aspekt funktioniert die Analyse bei Text-Dateien bei weitem wie der Unix-Befehl ”grep”. Es ist wichtig, an dieser Stelle darauf hinzuweisen, daß die Text-Dateien mit der Dateiextension **txt** entsprechend gekennzeichnet werden müssen, damit die Perl-Schnittstelle in Kenntnis gesetzt wird, um welchen Dateityp es sich dabei handelt. Zur Verdeutlichung der unterschiedlichen inhaltlichen Struktur wird nachfolgend jeweils ein Auszug aus einer Text-Datei und einer Words-Datei dargestellt:

Auszug Text-Datei

```
<{> [@] <|> [t] <|> [f] <|> [n] <}>
```

Sternzeit

```
29. März <[> 1997 <]> - Ein königlicher Blick auf die Milchstraße
```

Wir kennen Bilder von spiralförmigen Galaxien wie unserer Milchstraße. Sie erinnern an himmlische Strukturen, die majestätisch durch den Weltraum wirbeln.

Eine Spiralgalaxie hat kein ruhiges "Auge" in ihrer Mitte. Diese Mitte besteht aus einer runden Sternansammlung, die über und unter der flachen Spirale hervortritt.

Machen Sie sich ein Bild von einer Spiralgalaxie. Stellen Sie sich vor, Sie sind ein König und betrachten Ihr Reich von Ihrem Schloss aus, das sich auf einer Anhöhe befindet. Das Schloss ist auf allen Seiten von flachen Ebenen umgeben. Die Anhöhe ist die Sternensammlung in der Mitte der Milchstraße. Die umliegenden Ebenen bilden die Galaxienscheibe....

```
\subsubsection{Auszug Words--Datei}
\begin{verbatim}
0.040000 121 [0]
0.130000 121 [t]
0.200000 121 [t]
0.350000 121 [0]
0.440000 121 [t]
0.490000 121 [0]
0.530000 121 [n]
0.630000 121 [n]
0.710000 121 [0]
0.800000 121 [n]
0.860000 121 [0]
0.940000 121 [t]
1.020000 121 [0]
1.090000 121 [f]
1.220000 121 [0]
1.290000 121 [t]
1.510000 121 [0]
1.570000 121 [t]
1.710000 121 [0]
1.760000 121 [0]
1.820000 121 [t]
1.900000 121 [0]
1.930000 121 [n]
2.020000 121 [0]
2.070000 121 [0]
2.110000 121 [n]
```

Es ist ersichtlich, daß in der Words-Datei Abtastwerte in einem bestimmten Format gespeichert werden. Im Gegensatz dazu enthält die Textdatei lediglich Plantext.

Darüber hinaus muß auf jeden Fall sichergestellt werden, daß die gesamte Inputdatei (beispielsweise test.dat) ausschließlich Text-Dateien enthält. Auf diese Weise erhält die Ergebnisdatei ein einheitliches Format.

4.8 Bemerkungen zum Perl-Tools (Hilfsroutinen)

Perl-Tools (addpath.pl oder 2mssql.pl beispielsweise) sind kleine Perl-Programme, deren Namen die Dateiextension ".pl" noch tragen. Üblicherweise haben Unix-Befehle keine Extension. Unter diesem Aspekt können all die Perl-Programme (einschließlich des Analyseprogramms) umbenannt werden, indem die Endung entfällt.

Kapitel 5

Testphase

Um die Tauglichkeit und Leistungsfähigkeit des Programms richtig einschätzen zu können, soll sich das Programm unbedingt einer Testphase unterziehen. In diesem Kapitel wird u.a. das Testszenario beschrieben sowie das Testergebnis protokolliert.

5.1 Testszenario

Es handelt sich hierbei in erster Linie um die Zusammensetzung der Testdaten und die Operationen, die auf die Testdaten angewandt werden sollen. Der Testvorgang besteht im wesentlichen aus folgenden Hauptschritten:

- Bereitstellung einer leeren Sprachsignaldatenbank
- Ausfüllen der Testdatenbank mit Testdaten
- Löschen-Operation in Batchform testen
- Formatumwandlung testen
- Analyse durch die Perl-Schnittstelle

5.1.1 Bereitstellung einer leeren Sprachsignaldatenbank

Es wird eine neue Datenbank namens **korpora** erstellt mit allen Tabellen, die keine Daten enthalten. Es wird wie folgt vorgegangen:

- Datenbank "korpora" erzeugen: *mysql create korpora.*
- Einen Auszug von bisheriger Datenbank signal erzeugen: *mysqldump signal > signal.dmp.*
- Die Auszugsdatei signal.dmp modifizieren, indem alle darin enthaltenen INSERT-Statements löschen. Diese geänderte Auszugsdatei signal.dmp enthält nun SQL-Statements zur Erzeugung der Tabellenstruktur (wie z.B. CREATE), die sogenannten DDL-Statements (data definition language). Auf diese Datei kann immer zurückgegriffen werden, wenn eine leere Sprachsignal-Datenbank benötigt wird. Aus diesem Grund ist es ratsam, die Datei signal.dmp gut aufzuheben.
- Tabellenstruktur in die neue Datenbank "korpora" übertragen: *mysql korpora < signal.dmp.*

Im Laufe der Testphase kann der oben dargestellte Vorgang immer wiederholt werden, damit man eine leere Sprachsignaldatei-Datenbank zur Verfügung hat. Zu diesem Zweck kann eine ausführbare Datei namens **korpodb** (beispielsweise) erzeugt werden, die folgenden Inhalt hat:

```
mysqladmin drop korpora
mysqladmin create korpora
mysql korpora < signaltemplate.dmp
```

wobei die Datei `signaltemplate.dmp` die gesicherte Auszugsdatei ist, die nur die Definition der Tabellenstruktur enthält. Um eine leere Testdatenbank namens "korpora" zu erzeugen, braucht man nun nur `korpodb` einzugeben.

An dieser Stelle ist es angebracht, auf den Abschnitt 1.4.3 hinzuweisen, in dem eine ausführlichere Beschreibung vorliegt.

5.1.2 Ausfüllen der Testdatenbank mit Testdaten

Nun sollen Testdaten in die leere Datenbank "korpora" importiert werden. Es handelt sich dabei in erster Linie um Daten von Sprachsignaldateien, die in die Tabelle FILE übernommen werden sollen. Es wird von folgender Ausgangskonstellation ausgegangen, bei der die Sprachsignaldateien unter den Verzeichnissen vorliegen, die nachfolgernd aufgelistet sind:

- /graugans/cd/stern1/sd: Sd-Dateien
- /graugans/cd/stern1/phones: Phones-Dateien
- /graugans/cd/stern1/phonemics: Phonemics-Dateien
- /graugans/cd/stern1/texte: Text-Dateien
- /graugans/cd/stern1/html: Html-Dateien
- /graugans/cd/stern1/tags: Tags-Dateien
- /graugans/cd/stern1/syl : Syl-Dateien
- /graugans/cd/stern1/syllables: Syllables-Dateien
- /graugans/cd/stern1/words: Words-Dateien
- -----
- /graugans/cd/stern2/sd: Sd-Dateien
- /graugans/cd/stern2/phones: Phones-Dateien
- /graugans/cd/stern2/phonemics: Phonemics-Dateien
- /graugans/cd/stern2/texte: Text-Dateien
- /graugans/cd/stern2/html: Html-Dateien
- /graugans/cd/stern2/tags: Tags-Dateien
- /graugans/cd/stern2/syl : Syl-Dateien
- /graugans/cd/stern2/syllables: Syllables-Dateien

Nun gilt es zunächst, all diese Sprachsignaldateien nebst Pfadangabe in eine Inputdatei zu übernehmen, die als Input für die Einleseroutine dient. Zu diesem Zweck werden die Perl-Tools `addpath.pl` und `2mysql.pl` (siehe 6.3) herangezogen. Es wird für jedes Verzeichnis (und somit für jeden Dateityp) jeweils eine Inputdatei erzeugt, indem die zwei oben genannten Perl-Programme im Zusammenspiel mit dem Unix-Befehl "ls -l" eingesetzt werden. Dies geschieht für die Dateien unter dem Verzeichnis `/graugans/cd/stern1` wie folgt:

- Inputdatei *sd.in* für Verzeichnis */graugans/cd/stern1/sd*
 - *ls -l /graugans/cd/stern1/sd > sd.dat*
 - *perl addpath.pl sd.dat /graugans/cd/stern1/sd/ > sd.temp*
 - *perl 2msql.pl sd.temp > sd.in*
- Inputdatei *phones.in* für Verzeichnis */graugans/cd/stern1/phones*
 - *ls -l /graugans/cd/stern1/phones > phones.dat*
 - *perl addpath.pl phones.dat /graugans/cd/stern1/phones/ > phones.temp*
 - *perl 2msql.pl phones.temp > phones.in*
- Inputdatei *phonemics.in* für Verzeichnis */graugans/cd/stern1/phonemics*
 - *ls -l /graugans/cd/stern1/phonemics > phonemics.dat*
 - *perl addpath.pl phonemics.dat /graugans/cd/stern1/phonemics/ > phonemics.temp*
 - *perl 2msql.pl phonemics.temp > phonemics.in*
- Inputdatei *texte.in* für Verzeichnis */graugans/cd/stern1/texte*
 - *ls -l /graugans/cd/stern1/texte > texte.dat*
 - *perl addpath.pl texte.dat /graugans/cd/stern1/texte/ > texte.temp*
 - *perl 2msql.pl texte.temp > texte.in*
- Inputdatei *html.in* für Verzeichnis */graugans/cd/stern1/html*
 - *ls -l /graugans/cd/stern1/html > html.dat*
 - *perl addpath.pl html.dat /graugans/cd/stern1/html/ > html.temp*
 - *perl 2msql.pl html.temp > html.in*
- Inputdatei *tags.in* für Verzeichnis */graugans/cd/stern1/tags*
 - *ls -l /graugans/cd/stern1/tags > tags.dat*
 - *perl addpath.pl tags.dat /graugans/cd/stern1/tags/ > tags.temp*
 - *perl 2msql.pl tags.temp > tags.in*
- Inputdatei *syl.in* für Verzeichnis */graugans/cd/stern1/syl*
 - *ls -l /graugans/cd/stern1/syl > syl.dat*
 - *perl addpath.pl syl.dat /graugans/cd/stern1/syl/ > syl.temp*
 - *perl 2msql.pl syl.temp > syl.in*
- Inputdatei *syllables.in* für Verzeichnis */graugans/cd/stern1/syllables*
 - *ls -l /graugans/cd/stern1/syllables > syllables.dat*
 - *perl addpath.pl syllables.dat /graugans/cd/stern1/syllables/ > syllables.temp*
 - *perl 2msql.pl syllables.temp > syllables.in*
- Inputdatei *words.in* für Verzeichnis */graugans/cd/stern1/words*
 - *ls -l /graugans/cd/stern1/words > words.dat*
 - *perl addpath.pl words.dat /graugans/cd/stern1/words/ > words.temp*
 - *perl 2msql.pl words.temp > words.in*

Die Erstellung der Inputdateien für die Sprachsignaldateien unter dem Verzeichnis /graugans/cd/stern2 wird analog vorgenommen.

Nun wird das C-Programm mit der entsprechenden Kommandozeilenoption gestartet, so daß an die neu erstellte leere Datenbank "korpora" angebunden wird. Der Aufruf lautet in diesem Fall *korpo -s korpora*. Anschließend wird der Reihe nach Menüpunkt 3 (Datenbankroutinen eintragen), Menüpunkt 1 (Datei eintragen) und am Schluß Menüpunkt 2 (Einlesen von Dateien) ausgewählt. Dieser Einlesevorgang wiederholt sich für jede zuvor erstellte Inputdatei (sd.in, sd2.in, phones.in, phones2.in, phonemics.in, phonemis2.in, tags.in, tags2.in, words.in, words2.in, syl.in, syl2.in, syllables.in, syllables2.in, html.in, html2.in, texte.in, texte2.in), so daß alle Sprachsignaldateien in diesen Inputdateien in die Datenbank "korpora" übertragen werden können (siehe auch 1.4.3).

Wie in 3.4.14 beschrieben kann der Aufbau der Datenbank korpora auch mit den folgenden Kommandozeilen erfolgen:

- korpo -s korpora A=3 B=1 C=2 sd.in /graugans/cd/stern1/sd/ -e
- korpo -s korpora A=3 B=1 C=2 sd2.in /graugans/cd/stern2/sd/ -e
- korpo -s korpora A=3 B=1 C=2 phones.in /graugans/cd/stern1/phones/ -e
- korpo -s korpora A=3 B=1 C=2 phones2.in /graugans/cd/stern2/phones/ -e
- korpo -s korpora A=3 B=1 C=2 phonemics.in /graugans/cd/stern1/phonemics/ -e
- korpo -s korpora A=3 B=1 C=2 phonemics2.in /graugans/cd/stern2/phonemics/ -e
- korpo -s korpora A=3 B=1 C=2 tags.in /graugans/cd/stern1/tags/ -e
- korpo -s korpora A=3 B=1 C=2 tags2.in /graugans/cd/stern2/tags/ -e
- korpo -s korpora A=3 B=1 C=2 words.in /graugans/cd/stern1/words/ -e
- korpo -s korpora A=3 B=1 C=2 words2.in /graugans/cd/stern2/words/ -e
- korpo -s korpora A=3 B=1 C=2 syl.in /graugans/cd/stern1/syl/ -e
- korpo -s korpora A=3 B=1 C=2 syl2.in /graugans/cd/stern2/syl/ -e
- korpo -s korpora A=3 B=1 C=2 syllables.in /graugans/cd/stern1/syllables/ -e
- korpo -s korpora A=3 B=1 C=2 syllables2.in /graugans/cd/stern2/syllables/ -e
- korpo -s korpora A=3 B=1 C=2 html.in /graugans/cd/stern1/html/ -e
- korpo -s korpora A=3 B=1 C=2 html2.in /graugans/cd/stern2/html/ -e
- korpo -s korpora A=3 B=1 C=2 texte.in /graugans/cd/stern1/texte/ -e
- korpo -s korpora A=3 B=1 C=2 texte2.in /graugans/cd/stern2/texte/ -e

5.1.3 Löschen-Routine in Batchform testen

Es wird in diesem Fall eine neue Inputdatei für die Löschroutine erstellt. Zu diesem Zweck wird das Perl-Programm 2msql.pl so modifiziert, daß es eine Inputdatei erzeugt, die nur Informationen über NAME, PRE, PATH und EXT in der genannten Reihenfolge enthält. Zur Veranschaulichung wird das geänderte Programm 2msql.pl nachfolgend aufgelistet:

```
use File::Basename;

defined @ARGV || die "Falsche Syntax\n";

print "#! NAME PRE PATH EXT\n";
while ( $full = <> ) { chop $full;
  ( $name, $path, $suf ) = fileparse( $full, '\.[^\.]+' );
  $name =~ /(.*) (\d\d) (\d\d) (\d\d) ([\.\d+]*)/;
  $pre = $1 ? $1 : "\"\"";
  $date = "$4.$3.$2";
  #$time = $5 ? $5 : "\"\"";
  $temp = $5 ? $5 : "\"\"";
  if($temp ne "\"") {
    $temp =~ /(\.)(.*)/;
    $time = $2;
  }
  else
  {
    $time = $temp;
  }
  $ext = $suf; $ext =~ s/\.(.*)/$1/;
  print "$name$suf\t$time$pre\t$path\t$ext\n";
}
```

Nach Erstellung der Inputdatei wird die Löschroutine in Batchform durchs Aktivieren des entsprechenden Menüpunkts aufgerufen. Bei der Frage nach der Inputdatei wird die gerade erstellte Datei angegeben. Nach diesem Vorgang wird eine Abfrage gestartet, die alle sich in der Datenbank befindenden Dateien sammelt. Die in der Inputdatei aufgelisteten Dateien sind nicht mehr in der Datenbank vorhanden. Das Ergebnis ist also zufriedenstellend.

Mit Kommandozeile im vollständigen Automatisierungsmodus kann dieser Vorgang wie folgt vorgenommen werden:

```
korpo -s test A=4 B=2 "Inputdatei" -e
```

5.1.4 Formatumwandlung testen

Vor diesem Testvorgang wird die Datenbank "korpora" entsprechend präpariert, daß zumindest Sprachsignaldateien mit dem Sd-Format darin enthalten sind, weil die Transformationsprozeduren in der Regel auf Dateien solchen Formats angewandt werden. Die Formatumwandlung wird nach folgendem Szenario getestet:

- Formatumwandlung für einzelne Dateien (interaktiv): Es wird der Reihe nach der entsprechende Menüpunkt ausgewählt. Als Resultat gelten folgende Feststellungen:

- Die Umwandlung zum Sph-Format mittels der Transformationsprozedur e2sphere ist am schnellsten.
- Alle anderen Umwandlungsvorgänge mittels der Transformationsprozeduren Alignphones, Alignwords dauert eine ganze Weile (ca. 2 Stunde für eine einzige Sprachsignaldatei), da die Matrixdatei mit der Dateierweiterung Mfc temporär erzeugt werden muß.
- Formatumwandlung mit einer Inputdatei: Im Prinzip wird dabei ähnlich vorgegangen wie der interaktive Test. In diesem Fall werden aber eine Inputdatei namens "testformat.dat" erstellt, in der 5 Sprachsignaldateien im Sd-Format gespeichert sind. Es wird der Reihe die jeweilige Umwandlung durchgeführt. Hinsichtlich der Laufzeit gilt die gleiche Feststellung wie beim interaktiven Test.

Zur Veranschaulichung wird der Inhalt der Testdatei "testformat.dat" nachfolgend aufgelistet:

```
dlf970325.1656.sd
dlf970326.1656.sd
dlf970327.1656.sd
dlf970328.1656.sd
dlf970329.1656.sd
```

5.1.5 Perl-Schnittstelle testen

Wie bereits erwähnt wird eine Ergebnisdatei bei den meisten Abfragen erzeugt, die auf der Tabelle FILE basieren. Diese Ergebnisdatei dient wiederum als Inputdatei für die Perl-Schnittstelle. Allerdings muß die Datei zuvor entsprechend vorbereitet werden, indem das Perl-Programm "2mssl.pl" herangezogen und auf diese Datei angewandt wird. Die Outputdatei dieses Vorgangs wird als endgültige Inputdatei für die Perl-Schnittstelle verwendet. Das Testen der Perl-Schnittstelle erfolgt im wesentlichen nach folgendem Szenario:

- Per Anfrage eine Inputdatei erzeugen.
- Basierend auf dieser Inputdatei wird die Perl-Schnittstelle in drei Durchgängen getestet:
 - Suche nach Features mit Kommandozeileoption -F.
 - Suche nach Sprachelementen mit Kommandozeileoption -E.
 - Suche sowohl nach Features als auch nach Sprachelementen gleichzeitig.

Erzeugung von Inputdateien

Zur Erinnerung sind die betroffenen Inputdateien Ergebnisse derjenigen Abfragen, die auf der Tabelle FILE basieren. Vielmehr handelt es sich um Abfragen, die als Ergebnis eine Liste von Sprachsignaldateien liefern, die im für die Analyse bei der Perl-Schnittstelle geeignete Format vorliegen (Sd-, Phones-, Text- und Words-Format). Die Aktivierung der Abfrage erfolgt durch den Hauptmenüpunkt **A=1** im C-Programm. Dabei wird der Benutzer aufgefordert, den Namen der Ergebnisdatei anzugeben. Genau diese Ergebnisdatei dient als Inputdatei für die anschließende Analyse bei der Perl-Schnittstelle.

- *korpo -s korpora A = 1 B = 1 C = 9 < RETURN >* (entsprechenden Menüpunkt aktivieren)
- *phones < RETURN >* (nur Phones-Format berücksichtigen)

- *phones.dat* < RETURN > (Name der Ergebnisdatei)

Die Erstellung der Inputdatei *phones.dat* kann im vollständigen Automatisierungsmodus wie folgt vorgenommen werden:

- *korpo -s korpora A = 1 B = 1 C = 9 phones -e* < RETURN > (entsprechenden Menüpunkt
- In diesem Fall heißt die Outputdatei *erg.dat*, die nach *phones.dat* umbenannt werden muß. Dies geschieht auf Shell-Ebene wie folgt: *mv erg.dat phones.dat*.

Suche nach Features mit Kommandozeileoption -F

Die Features-Analyse kann beispielsweise bei Sprachsignaldateien vorgenommen werden, die "suchfähige" Features enthalten. Unter diesem Aspekt kommen Sprachsignaldateien mit den Formaten Phones und Syllables in Frage.

Bei Phones-Dateien Für die Features-bezogene Analyse wird die im ersten Schritt erzeugte Datei *phones.in* als Inputdatei verwendet, die insgesamt 57 Sprachsignaldateien enthält, welche ihrerseits vom Ordner */graugans/cd/Stern1/sd/* stammen. Die Aktivierung der Analyse lautet in diesem Fall wie folgt:

```
analyse.pl -v -F "vo|stl" -k defi.dat -pro "sgplay -s %S -e %E %F # %-1 %1 %I"
-r phones.out phones.dat
```

Dabei wird die Definitionsdatei *defi.dat* wie in 4.3 definiert, deren zufolge nach den Features a, e, i, p, f und t gesucht wird. Das Ergebnis wird in der Datei *phones.out* gespeichert.

Bei Syllables-Dateien Da die Syllables-Dateien eine ähnliche inhaltliche Struktur wie die Phones-Dateien besitzen, kann die Analyse per Kommandoswitch [-F] auch bei Syllables-Dateien erfolgen. Es wird zunächst die Inputdatei *syllables.dat* erstellt, die wie gewöhnlich das Ergebnis folgender Abfrage ist:

- *korpo -s korpora A = 1 B = 1 C = 9* < RETURN > (entsprechenden Menüpunkt aktivieren)
- *syllables* < RETURN > (nur Syllables-Format berücksichtigen)
- *syllables.dat* < RETURN > (Name der Ergebnisdatei)

Die Datei *syllables.dat* wird nun verwendet für die Analyse, die wie folgt gestaltet ist:

```
analyse.pl -v -F "na" -k defi.dat -pro "sgplay -s %S -e %E %F %I"
-r syllables.out syllables.dat
```

Die Erstellung der Inputdatei *syllables.dat* kann im vollständigen Automatisierungsmodus wie folgt vorgenommen werden:

- *korpo -s korpora A = 1 B = 1 C = 9 syllables -e* < RETURN > (entsprechenden Menüpunkt
- Umbenennung der voreingestellten Outputdatei *erg.dat* nach *syllables.dat*: *mv erg.dat syllables.dat*

Suche nach Elementen mit Kommandozeileoption -E

Im Vergleich zur Feature-basierten Analyse kann die Elementen-basierte Analyse aus programmatischer Sicht bei allen Sprachsignaldateien vorgenommen werden, in denen Elemente als Wort aufgefaßt werden kann. D.h, ein Feature kann auch als ein Wort ausgelegt werden. Aus linguistischer Sicht wird die Element-basierte Analyse (oder segment-basierte Analyse) allerdings bei Sprachsignaldateien durchgeführt werden, die charakteristisch Wörter enthalten. Beispiele für diese Art der Sprachsignaldateien sind Words-, Text- und Tags-Dateien.

Bei Words-Dateien Für die Elementen-bezogene Analyse wird eine neue Inputdatei namens *words.in* erzeugt, in der alle Sprachsignaldateien mit dem Words-Format vom Ordner /graugans/cd/Stern1/words aufgelistet sind. Diese Datei basiert auf einer Abfrage, die wie folgt erstellt wird:

- *korpo -s korpora A = 1 B = 1 C = 9 < RETURN >* (entsprechenden Menüpunkt aktivieren)
- *words < RETURN >* (nur Words-Format berücksichtigen)
- *words.dat < RETURN >* (Name der Ergebnisdatei)

Die Aktivierung der Analyse lautet in diesem Fall wie folgt:

```
analyse.pl -v -E "durch" -r words.out words.dat
```

Die Erstellung der Inputdatei *words.dat* kann im vollständigen Automatisierungsmodus wie folgt vorgenommen werden:

- *korpo -s korpora A = 1 B = 1 C = 9 words -e < RETURN >* (entsprechenden Menüpunkt aktivieren)
- Umbenennung der voreingestellten Outputdatei *erg.dat* nach *words.dat*: `mv erg.dat words.dat`

Bei Tags-Dateien Die Suche nach Sprachelementen per Kommandozeileoption [-E] kann nicht nur in den Words-, sondern auch in den Text- und Tags-Dateien vorgenommen werden. Im nächsten Abschnitt wird demonstriert, wie die Text-Dateien analysiert werden. Nun folgt ein zweiter Test der Kommandozeileoption [-E], der nach folgendem Schema vorgegangen wird:

- *korpo -s korpora A = 1 B = 1 C = 8 < RETURN >* (entsprechenden Menüpunkt aktivieren)
- *tags < RETURN >* (nur Tags-Format berücksichtigen)
- *tags.dat < RETURN >* (Name der Ergebnisdatei)

Die Aktivierung der Analyse lautet in diesem Fall wie folgt:

```
analyse.pl -v -E ''APPR'' -r tags.out tags.dat
```

Die Erstellung der Inputdatei *tags.dat* kann im vollständigen Automatisierungsmodus wie folgt vorgenommen werden:

- *korpo -s korpora A = 1 B = 1 C = 9 tags -e < RETURN >* (entsprechenden Menüpunkt aktivieren)
- Umbenennung der voreingestellten Outputdatei *erg.dat* nach *tags.dat*: `mv erg.dat tags.dat`

Bei Text-Dateien Die Analyse bei Text-Dateien kann als Sonderfall betrachtet werden, weil sie in der Regel nur Texte enthalten und deren strukturelle Aufbau keine Angabe über Start- und Endzeit eines Sprachelements aufweist. Dementsprechend gilt in diesem besonderen Fall die Voreinstellung nicht, bei der das Programm "sgplay" samt Start- und Endzeit automatisch in die Outputdatei aufgenommen wird, sofern diesbezüglich keine ausdrückliche Angabe erfolgt. Überdies hat die Kommandozeileoption [-pro] in diesem Fall aber keine Auswirkung. Der voreingestellte Output besitzt unter anderem folgenden Aufbau:

- die Textzeile, wo das gesuchte Wort gefunden wird,
- den ursprünglichen Dateinamen (Dateiextension wird nicht automatisch nach SD-Format umbenannt),
- und das gesuchte Wort.

Die drei obengenannten Daten werden als String zusammengefaßt und Zeile für Zeile in die Outputdatei geschrieben. Somit enthält die Outputdatei bei Textdateien Informationen darüber, in welcher Datei und bei welcher Zeile sich das gesuchte Wort befindet.

Die Aktivierung der Analyse lautet in diesem Fall wie folgt:

```
analyse.pl -v -E '(Stern)|Mitte' -r text.out text.dat
```

Suche nach Elementen mit Kommandozeileoptionen -F und -E

Bei diesem Analyseverfahren werden die Kommandozeileoptionen -F und -E miteinander kombiniert. Es wird eine neue Inputdatei benötigt, in der Sprachsignaldateien enthalten sind, bei denen sowohl nach Features als auch nach Elementen gesucht werden kann. Unter diesem Aspekt ist die neue Inputdatei eine Kombination aus zwei Dateien, nämlich der zuvor erstellten Datei phones.dat und der Datei text.dat.

Die Datei text.dat ist wiederum das Ergebnis der Anfrage über Txt-Dateien in der Datenbank korpora. Zum Testen werden zuvor alle Txt-Dateien unter dem Verzeichnis /graugans/cd/stern1/texte in die Datenbank übertragen. Aus diesem Grund enthält die Datei text.dat alle Sprachsignaldateien unter /graugans/cd/stern1/texte. Die Erstellung der Datei text.dat geschieht wie folgt:

- *korpo -s korpora A = 1 B = 1 C = 9 < RETURN >* (entsprechenden Menüpunkt aktivieren)
- *txt < RETURN >* (nur Txt-Format berücksichtigen)
- *text.dat < RETURN >* (Name der Ergebnisdatei)

Die Erstellung der Inputdatei text.dat kann im vollständigen Automatisierungsmodus wie folgt vorgenommen werden:

- *korpo -s korpora A = 1 B = 1 C = 9 text -e < RETURN >* (entsprechenden Menüpunkt)
- Umbenennung der voreingestellten Outputdatei erg.dat nach text.dat: `mv erg.dat text.dat`

Wie bereits erwähnt besteht die neue Inputdatei aus phones.dat und text.dat. Die Erstellung der neuen Inputdatei namens kombi.dat kann wie folgt von statten gehen:

```
cat phones.dat text.dat > kombi.dat
```

Nun wird die Analyse mittels folgendes Befehls gestartet:

```
analyse.pl -v -F "vo|stl" -E "durch" -k defi.dat -pro "sgplay -s %S -e %E %-1 %1
%F %I"
-r kombi.out kombi.dat
```

Danach wird die Analyse der Frequenz der gesuchten Elementen und Features genau nach der gleichen Methode wie oben beschrieben angeschlossen.

5.1.6 Auswertung der Analyse bei der Perl-Schnittstelle

Nun wird untersucht, ob das Ergebnis der Analyse stimmt, indem die Anzahl der gefundenen Elementen in der jeweiligen Ergebnisdatei sowie die Anzahl genau dieser Elementen in den Quelldateien berechnet wird. Diese zwei Größen werden miteinander verglichen. Es gibt dafür zwei Möglichkeiten: mit Hilfe von Shellskripts und mit Hilfe des Perl-Tools `checkele.pl`.

Mit Hilfe von Shellskripts

In diesem Fall werden die Shellskripts `woerter` und `freq` verwendet (siehe 6.2).

Bei Phones-Dateien Vorweggenommen wird im folgenden detailliert beschrieben, wie die Analyseauswertung mittels Shellskripts vorgenommen wird. Bei Ergebnisdateien anderer Formate (Words, Tags, Text, Syllables) geht die Analyse analog von statten.

Mit dem Shellskript `woerter` wird eine temporäre Datei erzeugt, in der alle in der Datei `phones.out` vorkommenden Wörter zeilenweise aufgelistet. Angewandt auf jene temporäre Datei erzeugt das Shellskript seinerseits eine andere Datei, in der aufgezählt wird, wie oft die Wörter vorkommen. Dies geschieht wie folgt:

- Ermittlung der Wörterfrequenz in der Datei `phones.out`
 - `woerter phones.out > phones_1.out`
 - `freq phones_1.out > phones_2.out`
- Ermittlung der Wörterfrequenz zum Vergleich in den Quelldateien

In diesem Fall werden die zwei Skripts `woerter` und `freq` auf jede Sprachsignaldatei angewandt, in der gesucht wird. Dies sind genau diejenige Dateien, die in der Datei **phones.dat** spezifiziert sind. Am Ende dieser Prozedere enthält man für jede Sprachsignaldatei aus der Datei `phones.dat` eine temporäre Datei, in der die Anzahl der darin vorkommenden Wörter manifestiert ist, bei deren Erstellung analog zur Ermittlung der Wörterfrequenz in der Datei `phones.out` vorgegangen wird. Dies geschieht wie folgt:

 - `cat /graugans/cd/stern1/phones/* > phonesver.dat`
 - `woerter phonesver.dat > phonesver_1.dat`
 - `freq phonesver_1.dat > phonesver_2.dat`
- Vergleich

Für jedes gesuchte Sprachelement wird dessen gesamte Anzahl in den Sprachsignaldateien aufsummiert und mit der ermittelten Anzahl im Schritt 1 verglichen. Bei diesem Vergleich kann das Skript `hgrep` nützlich sein. Dieses Skript funktioniert wie den Unix-Befehl `grep`, nur mit dem Unterschied, daß die Stellen der Hits hervorgehoben dargestellt werden (siehe 6.2). Im folgenden wird das Vergleichsergebnis bei der Suche nach dem Feature `"t"` illustriert:

 - 36677 `"t"` in `phonesver_2.dat`
 - 36677 `"t"` in `phones_2.out`

Bei Syllables-Dateien Die Ergebnisdatei `syllables.out` wird durch Anwendung der Shellskripts `woerter` und `freq` in zwei Dateien `syllables_1.out` und `syllables_2.out` überführt. Es wird dann die zuvergleichende Datei mit dem Befehl `cat /graugans/cd/stern1/syllables/* > syllablesver.dat` erzeugt, die anschließend durch Anwendung von `woerter` und `freq` ebenfalls in die Datei `syllables_1.dat` und `syllables_2.dat` überführt wird. Mittels der Befehle `grep` und/oder `hgrep` kann die Frequenz der zu analysierenden Sprachelemente in den zwei Dateien `syllables_2.out` und `syllablesver_2.dat` ermittelt und miteinander verglichen werden. Zur Illustration wird das Vergleichsergebnis nachfolgend dargestellt:

- 21377 "n" in `syllables_2.out`
- 21376 "n" und 1 "N" in `syllablesver_2.dat`

Bei Words-Dateien Zur Auswertung des Analyseergebnisses wird auch wie oben beschrieben vorgegangen. Es entsteht folgendes Ergebnis:

- 21377 "n" in `syllablesref.out`
- 21377 "n" in `syllables_n.dat`

Es entstehen in diesem Fall die Dateien `words_1.out` (nach Anwendung von `woerter`) und `words_2.out` (nach Anwendung von `freq`). Als Vergleichsdatei wird die Datei `wordsver_2.dat` verwendet, die wie folgt erstellt wird:

- `cat /graugans/cd/stern1/words/* > wordsver.dat`
- `woerter wordsver.dat > wordsver_1.dat`
- `freq wordsver_1.dat > wordsver_2.dat`

Das Vergleichsergebnis sieht in diesem Fall wie folgt aus:

- 36 "durch" in `words_2.out`
- 31 "durch" und 5 "Durch" in `wordsver_2.dat`

Bei Tags-Dateien In diesem Fall wird in allen in der Datei `tags.dat` aufgelisteten Tags-Dateien nach dem Wort "APPR" gesucht. Es findet anschließend wie üblich mittels der Skripten die Ermittlung der Frequenz von "APPR" statt. Die Ergebnisdateien heißen nun `tags_1.out` und `tags_2.out`. Als Vergleichsdatei wird die Datei `tagsver_2.dat` verwendet, die wie folgt erstellt wird:

- `cat /graugans/cd/stern1/tags/* > tagsver.dat`
- `woerter tagsver.dat > tagsver_1.dat`
- `freq tagsver_1.dat > tagsver_2.dat`

Das Vergleichsergebnis sieht in diesem Fall wie folgt aus:

- 2218 "APPR" in `tags_2.out`
- 2218 "APPR" in `tagsver_2.dat`

Bei Kombi-Dateien In diesem Fall besteht die gesamte Inputdatei aus den zwei Dateien text.dat und phones.dat. Dementsprechend beinhaltet die Vergleichsdatei all die Dateien, die sowohl in text.dat als auch in phones.dat enthalten sind, und wir wie folgt erstellt:

- `cat /graugans/cd/stern1/phones/ * /graugans/cd/stern1/texte/* > kombiver.dat`
- `woerter kombiver.dat > kombiver_1.dat`
- `freq kombiver_1.dat > kombiver_2.dat`

Mittels `hgrep` und/oder `grep` wird in den zwei Ergebnisdateien `kombi_2.out` und `kombiver_2.dat` gesucht und deren Anzahl miteinander verglichen.

Bei Text-Dateien Wie bereits erwähnt verläuft die Analyse bei Text-Dateien ganz anders. Zur Auswertung der Ergebnisse in diesem Fall eignet sich das Analyseverfahren mit Hilfe von Shellskripts nicht. Dazu soll man sich den im nächsten Abschnitt dargestellten Perl-Tools bedienen.

Mit Hilfe von Perl-Tools `checkele.pl` und `checkelege.pl`

Zur Erinnerung erfolgt die Suche bei der Analyse wortweise (mit einem regulären Ausdruck). Beispielsweise werden "durch" und "dadurch" als zwei verschiedene Elemente betrachtet. Bei `grep` und `hgrep` hat man aber keine Möglichkeit, einen regulären Ausdruck als Suchmuster zu spezifizieren, um gezielt ein ganzes Wort zu suchen. Aufgrund dieser Einschränkung wird das Perl-Tool `checkele.pl` erstellt, das in der Lage ist, ein ganzes Wort zu suchen und dessen Vorkommen in einer Datei zusammenzählt und diese Größe als Ergebnis zurückliefert. Der Aufruf lautet dieses Perl-Tools lautet **`checkele.pl <Suchelement> <Dateiname>`**. Dabei spezifiziert das Argument `<Dateiname>` die Outputdatei, die unmittelbar bei der Analyse durch das Programm `analyse.pl` erzeugt wird. Außerdem wird das Perl-Tool `checkelege.pl` erstellt, die für die Ermittlung der Frequenz der Elemente in den Quelldateien bestimmt ist. Dabei sind die Quelldateien diejenige Dateien, in denen die Suche nach einem bestimmten Sprachelement und/oder einem bestimmten Muster erfolgt wird. Dem Tool `checkelege.pl` wird als Argument die Inputdatei übergeben, in der die zu suchenden Dateien aufgelistet sind. Dementsprechend ist diese Inputdatei genau die Ergebnisdatei einer Anfrage, die das C-Programm zurückliefert (A=1B=1C=8) und wird bereits während der Testphase erzeugt (nämlich `phones.dat`, `words.dat`, `syllables.dat`, `text.dat`, `tags.dat`). Der Aufruf von `checkelege.pl` hat folgende Syntax: **`checkelege.pl <Suchelement> <Suchdatei> <Outputdatei>`**. Das dritte Argument `<Outputdatei>` ist optional. Wenn eine Outputdatei nicht angegeben ist, enthält als Voreinstellung die Datei namens `out.dat` das Ergebnis.

Nachfolgend wird der Vergleich anhand von `checkele.pl` mit jeweils einem bestimmten Suchmuster zur Demonstration etwas ausführlicher beschrieben.

Bei Phones-Dateien

- `checkele.pl "t" phones.out > phonesref.out`
- `checkelege.pl "t" phones.dat phones_t.dat`
- Ergebnis: 366677 in beiden Fällen

Bei Syllables-Dateien

- `checkele.pl "n" syllables.out > syllablesref.out`
- `checkelege.pl "n" syllables.dat syllables_n.dat`
- Ergebnis: 24279 in beiden Fällen

Bei Words-Dateien

- `checkele.pl "durch" words.out > wordsref.out`
- `checkelege.pl "durch" words.dat words_durch.dat`
- Ergebnis: 36 in `wordsref.out`, 36 in `words_durch.dat`

Bei Tags-Dateien

- `checkele.pl "APPR" tags.out > tagsref.out`
- `checkelege.pl "APPR" tags.dat tags_appr.dat`
- Vergleichsergebnis : 2218 in beiden Fällen

Bei Text-Dateien

- `checkele.pl "Stern" text.out > textref.out`
- `checkelege.pl "Stern" text.dat text_stern.dat`
- Vergleichsergebnis : 48 bei `textref.out`, 47 bei `text_stern.dat` (siehe 4.7.3)

Bei Kombi-Dateien

- `checkele.pl "t" kombi.out > kombiref.out`
- `checkelege.pl "t" kombi.dat kombi_t.dat`
- Ergebnis: 36797 bei `kombiref.out` , 36796 bei `kombi.dat` (siehe 4.7.3)

Kapitel 6

Anhang

In diesem Anhang werden in erster Linie die wichtigen C-Routinen zusammen mit dem jeweiligen Verweis auf ihre Anwendung aufgeführt. Darüber hinaus werden die verwendeten Perl-Tools aufgelistet. Es wird ebenfalls ausführlich beschrieben, wie diese Perl-Tools eingesetzt werden können.

6.1 Funktionenliste des C-Programms

In diesem Abschnitt werden die wichtigen Routinen aufgelistet, die in den einzelnen Modulen implementiert sind.

6.1.1 Modul `main.c`

- `int ab_file_umwandeln(int dbsock,int argc,char** argv)` (Seite 41)
- `void ab_file_alle(int dbsock,int exittag)` (Seite 65)
- `void ab_file_zugeordnet(int dbsock,int exittag)` (Seite 65)
- `void ab_file_haupttyp_zugeordnet(int dbsock,int exittag)` (Seite 69)
- `void ab_file_untertyp_zugeordnet(int dbsock, int exittag)` (Seite 66)
- `void ab_file_format_zugeordnet(int dbsock, int exittag)` (Seite 66)
- `void ab_file_name_auswahl(int dbsock, int argc, char** argv)` (Seite 66)
- `void ab_file_datum_auswahl(int dbsock, int argc, char** argv)` (Seite 67)
- `void ab_file_endung_auswahl(int dbsock,int argc, char** argv)` (Seite 67)
- `void ab_file_quelle_auswahl(int dbsock,int argc, char** argv)` (Seite 68)
- `void ab_file_pre_auswahl(int dbsock,int argc, char** argv)` (Seite 67)
- `void ab_file_pfad_auswahl(int dbsock,int argc, char** argv)` (Seite 68)
- `void ab_file_ouput(int dbsock,int argc, char** argv)` (Seite 68)
- `void ab_quelle_alle(int dbsock,int exittag)` (Seite 74)
- `void ab_quelle_zugeordnet(int dbsock,int exittag)` (Seite 75)
- `void ab_quelle_name_auswahl(int dbsock,int argc, char** argv)` (Seite 75)

- void ab_prozedur_alle(int dbsock, int exittag) (Seite 69)
- void ab_prozedur_zugeordnet(int dbsock, int exittag) (Seite 69)
- void ab_prozedur_name_auswahl(int dbsock,int argc, char** argv) (Seite 71)
- void ab_prozedur_typ_auswahl(int dbsock, int argc, char** argv) (Seite 71)
- void ab_format_alle(int dbsock,int exittag) (Seite 72)
- void ab_format_zugeordnet(int dbsock,int exittag) (Seite 72)
- void ab_format_name_auswahl(int dbsock,int argc, char** argv) (Seite 72)
- void ab_haupttyp_alle(int dbsock, int exittag) (Seite 72)
- void ab_haupttyp_zugeordnet(int dbsock, int exittag) (Seite 73)
- void ab_haupttyp_name_auswahl(int dbsock, int argc, char** argv) (Seite 73)
- void ab_untertyp_alle(int dbsock, int exittag) (Seite 73)
- void ab_untertyp_zugeordnet(int dbsock, int exittag) (Seite 74)
- void ab_untertyp_name_auswahl(int dbsock, int argc, char** argv) (Seite 74)
- void ab_zugriff_alle(int dbsock, int argc) (Seite 75)
- void ab_zugriff_zugeordnet(int dbsock, int argc) (Seite 76)
- void ab_filext_prozedur_zu(int dbsock, int argc, char** argv) (Seite 69)
- void ab_filext_prozedur_zu_ziel(int dbsock, int argc, char** argv) (Seite 70)
- int db_connect(char* dbname)
- void main(int argc,char** argv)

6.1.2 Modul `hilfe1.c`

- int get_act_ID(char* table,int* actID,int dbsock)
- int update_act_ID(char* table,int actID,int dbsock)
- void ermitteln_ID(char* table, char* daten, int* ID,int dbsock)
- void file_datum_eingabe_invers(char* r_datum)
- void file_pre_eingabe(char* r_pre)
- void file_datum_eingabe(char* r_datum)
- void file_endung_eingabe(char* r_endung)
- void file_name_eingabe(char* r_name)
- void file_verzeichnis_eingabe(char* r_ver)
- void file_uhrzeit_eingabe(char* r_uhrzeit)
- void file_quelle_eingabe(char* r_quelle)

- void db_prozedur_eintragen(int dbsock,int argc, char** argv) (Seite 46)
- void db_format_eintragen(int dbsock,int argc, char** argv) (Seite 47)
- void db_quelle_eintragen(int dbsock,int argc, char** argv) (Seite 49)
- void db_file_eintragen(int dbsock,int argc, char** argv) (Seite 76)
- void db_haupttyp_eintragen(int dbsock,int argc, char** argv) (Seite 48)
- void db_untertyp_eintragen(int dbsock,int argc, char** argv)(Seite 48)
- void db_file_prozedur_zuordnen(int dbsock,int argc, char** argv) (Seite 46)
- void db_file_zugriff_zuordnen(int dbsock,int argc, char** argv)(Seite 49)
- void db_file_haupttyp_zuordnen(int dbsock,int argc, char** argv) (Seite 48)
- void db_file_untertyp_zuordnen(int dbsock,int argc, char** argv) (Seite 48)
- void db_file_format_zuordnen(int dbsock,int argc, char** argv) (Seite 47)
- void db_file_quelle_zuordnen(int dbsock,int argc, char** argv) (Seite 50)
- void db_fileext_prozedur_zuordnen(int dbsock,int argc, char** argv) (Seite 46)

6.1.3 Modul `hilfe2.c`

- int test_digit(char* daten)
- void db_quelle_loeschen(int dbsock, int argc, char** argv) (Seite 53)
- void db_format_loeschen(int dbsock, int argc, char** argv) (Seite 52)
- void db_haupttyp_loeschen(int dbsock, int argc, char** argv) (Seite 52)
- void db_untertyp_loeschen(int dbsock, int argc, char** argv) (Seite 53)
- void db_prozedur_loeschen(int dbsock, int argc, char** argv) (Seite 52)
- void db_file_loeschen(int dbsock, int argc, char** argv) (Seite 50)
- void db_file_format_zuordnung_loeschen(int dbsock, int argc, char** argv) (Seite 77)
- void db_file_prozedur_zuordnung_loeschen(int dbsock, int argc, char** argv) (Seite 78)
- void db_file_haupttyp_zuordnung_loeschen(int dbsock, int argc, char** argv) (Seite 79)
- void db_file_untertyp_zuordnung_loeschen(int dbsock, int argc, char** argv) (Seite 79)
- void db_file_quelle_zuordnung_loeschen(int dbsock, int argc, char** argv) (Seite 80)
- void db_file_zugriff_zuordnung_loeschen(int dbsock, int argc, char** argv) (Seite 80)
- void db_format_zuordnung_loeschen(int dbsock, int argc, char** argv) (Seite 53)
- void db_prozedur_zuordnung_loeschen(int dbsock, int argc, char** argv) (Seite 54)
- void db_haupttyp_zuordnung_loeschen(int dbsock, int argc, char** argv) (Seite 56)
- void db_untertyp_zuordnung_loeschen(int dbsock, int argc, char** argv) (Seite 56)

- void db_quelle_zuordnung_loeschen(int dbsock, int argc, char** argv) (Seite 57)
- void db_zugriff_zuordnung_loeschen(int dbsock, int argc, char** argv) (Seite 57)
- void db_fileext_prozedur_zu_loeschen(int dbsock, int argc, char** argv) (Seite 55)
- void db_fileext_prozedur_zu_loeschen_ziel(int dbsock, int argc, char** argv) (Seite 55)
- void delete_tabelle_allgemein(int dbsock, int argc, char** argv) (Seite 58)

6.1.4 Modul `hilfe3.c`

- void db_file_einlesen_haupt(dbsock,op,myfunction,argc,argv)
 - int dbsock
 - char* op
 - int (*myfunction)(FILE*,int,int,char**,char*)
 - int argc
 - char** argv
 (Seite 51, 77)
- int db_file_read_file(FILE* fp,int dbsock,int spalte,char** r_spalte,char* quelle) (Seite 77)
- int db_file_delete_file(FILE* fp,int dbsock,int spalte,char** r_spalte,char* quelle) (Seite 51)

6.1.5 Modul `hilfe4.c`

- void db_change_format(int dbsock, int argc, char** argv) (Seite 58)
- void db_change_haupttyp(int dbsock, int argc, char** argv) (Seite 59)
- void db_change_untertyp(int dbsock, int argc, char** argv) (Seite 59)
- void db_change_prozedur(int dbsock, int argc, char** argv) (Seite 85)
- void db_change_prozedur_namen(int dbsock, int argc, char** argv) (Seite 86)
- void db_change_prozedur_typ(int dbsock, int argc, char** argv) (Seite 86)
- void db_change_quelle(int dbsock, int argc, char** argv) (Seite 59)
- void db_change_file(int dbsock, int argc, char** argv) (Seite 81)
- void db_change_file_namen(int dbsock, int argc, char** argv) (Seite 82)
- void db_change_file_stamm(int dbsock, int argc, char** argv) (Seite 82)
- void db_change_file_datum(int dbsock, int argc, char** argv) (Seite 83)
- void db_change_file_uhrzeit(int dbsock, int argc, char** argv) (Seite 83)
- void db_change_file_endung(int dbsock, int argc, char** argv) (Seite 84)
- void db_change_file_pfad(int dbsock, int argc, char** argv) (Seite 84)
- void db_change_file_quelle(int dbsock, int argc, char** argv) (Seite 85)
- void db_change_format_zuordnung(int dbsock, int argc, char** argv) (Seite 60)

- void db_change_haupttyp_zuordnung(int dbsock, int argc, char** argv) (Seite 61)
- void db_change_untertyp_zuordnung(int dbsock, int argc, char** argv) (Seite 62)
- void db_change_quelle_zuordnung(int dbsock, int argc, char** argv) (Seite 63)
- void db_change_prozedur_zuordnung(int dbsock, int argc, char** argv) (Seite 61)
- void db_change_zugriff_zuordnung(int dbsock, int argc, char** argv) (Seite 64)
- void db_change_file_format_zuordnung(int dbsock, int argc, char** argv) (Seite 89)
- void db_change_file_prozedur_zuordnung(int dbsock, int argc, char** argv) (Seite 90)
- void db_change_file_haupttyp_zuordnung(int dbsock, int argc, char** argv) (Seite 90)
- void db_change_file_untertyp_zuordnung(int dbsock, int argc, char** argv) (Seite 91)
- void db_change_file_quelle_zuordnung(int dbsock, int argc, char** argv) (Seite 92)
- void db_change_file_zugriff_zuordnung(int dbsock, int argc, char** argv) (Seite 92)
- void db_change_file_zuordnung(int dbsock, int argc, char** argv) (Seite 86)
- void db_change_file_format_zuordnung_mID(int ID,int dbsock) (Seite 86)
- void db_change_file_prozedur_zuordnung_mID(int ID,int dbsock) (Seite 86)
- void db_change_file_haupttyp_zuordnung_mID(int ID,int dbsock) (Seite 86)
- void db_change_file_untertyp_zuordnung_mID(int ID,int dbsock) (Seite 86)
- void db_change_file_quelle_zuordnung_mID(int ID,int dbsock) (Seite 86)
- void db_change_file_zugriff_zuordnung_mID(int ID,int dbsock) (Seite 86)
- void db_change_fileext_prozedur_zu_pro(int dbsock, int argc, char** argv) (Seite 93)
- void db_change_fileext_prozedur_zu_ext(int dbsock, int argc, char** argv) (Seite 94)
- void db_change_fileext_prozedur_zu_zielext(int dbsock, int argc, char** argv) (Seite 94)

6.1.6 Modul **hilfe5.c**

- void ab_file_anzahl_stamm(int dbsock, int argc, char** argv) (Seite 95)
- void ab_file_anzahl_datum(int dbsock, int argc, char** argv) (Seite 96)
- void ab_file_anzahl_uhrzeit(int dbsock, int argc, char** argv) (Seite 96)
- void ab_file_anzahl_endung(int dbsock, int argc, char** argv) (Seite 96)
- void ab_file_anzahl_quelle(int dbsock, int argc, char** argv) (Seite 97)
- void ab_file_anzahl(int dbsock, int exittag) (Seite 95)

6.1.7 Modul `hilfe6.c`

- `void ab_file_umwandeln_haupt(int dbsock, char *output, int argc, char** argv)` (Seite 41, 42, 42)
- `void ab_file_umwandeln_mnamen(int dbsock, char* file_name, char* output, int interaktiv)` (Seite 41, 42, 42)
- `int ab_file_umwandeln(int dbsock)` (Seite 41)
- `void ab_file_umwandeln_endung(int dbsock, char*output, int argc, char** argv)` (Seite 43)
- `int db_file_out_eintragen(int dbsock, char* file_name, char* 0 out_name, char* out_ext, char* ziel_pfad)`
- `int ab_file_umwandeln_sph(int dbsock, int argc, char** argv)` (Seite 42)
- `int ab_file_umwandeln_phones(int dbsock, int argc, char** argv)` (Seite 42)
- `int ab_file_umwandeln_words(int dbsock, int argc, char** argv)` (Seite 43)
- `int db_hauptfile_out_eintragen(int dbsock, char* name, char* pre, char* datum, char* uhrzeit, char* endung, char* pfad, char* quelle)`

6.1.8 Modul `hilfe7.c`

- `int A_antwort(int argc, char** argv, char** A_ebene)`
- `int A_B_antwort(int argc, char** argv, char** B_ebene)`
- `int AB_C_antwort(int argc, char** argv, char** C_ebene)`
- `int ABC_D_antwort(int argc, char** argv, char** D_ebene)`

6.2 Shellskripts, Perl-Tool und Dateien bei der Testphase

6.2.1 Shellskripts

Dieser Abschnitt gibt einen Überblick über die nützlichen Shellskripts, die bei der Auswertung der Analyse mit der Perl-Schnittstelle zur Verwendung kommen:

- **woerter**

Beschreibung:

Jede Zeile wird in einzelne Woerter zerlegt, und jedes Wort wird auf einer eigenen Zeile ausgegeben

Usage: *woerter Inputdatei > Outputdatei*

Beispiel: *woerter phones.out > ergebnis*

- **feld**

Beschreibung:

Eine beliebige Spalte der Inputdatei wird extrahiert

Usage: *feld Spaltennummer Inputdatei > Outputdatei*

Beispiel: *feld 8 phones.out > ergebnis*

- **freq**

Beschreibung:

Liefert die Anzahl der Wörter zurück, die zeilenweise angeordnet sind. Es ist nützlich beim Zusammenspiel mit dem Skript "woerter".

Usage: *freq Inputdatei > Outputdatei*

Beispiel: *freq phones.out > ergebnis*

- **hgrep**

Beschreibung:

Funktioniert wie "grep", mit dem Unterschied, daß die gesuchten Stellen hervorgehoben werden.

Usage: *hgrep gesuchtesWort Inputdatei > Outputdatei*

6.2.2 Perl-Tools checkele.pl und checkelege.pl zur Auswertung der Analyse

checkele.pl

```
#!/usr/local/bin/perl
use File::Basename;
local($ele);
local($input);
local($counter);

&usage unless @ARGV;

&usage if($#ARGV < 1);

$input = @ARGV[1];
$ele = @ARGV[0];
$counter = 0;

open IN, "<$input" or die "input nicht vorhanden";
while ( $full = <IN> ) {
    chop $full;
    $erg = grep(/\b$ele\b/i,$full);
    if($erg != 0) {
        $counter = $counter +1;
        print $full . " " . $counter . "\n";
    }
}

print "Anzahl der Elemente " . $counter;

sub usage()
{
    print "checkele.pl <Suchelement> <Dateiname> \n";
    exit;
}
```

checkelege.pl

```

#!/usr/local/bin/perl
use File::Basename;
local($ele);
local($input);
local($counter);
local ($gcounter);
local ($filename);
local ($output);
local($col1,$col2,$col3);

&usage unless @ARGV;

&usage if($#ARGV < 1);

$input = @ARGV[1];
$ele = @ARGV[0];
$output = @ARGV[2];

$counter = 0;
$gcounter = 0;

open IN, "<$input" or die "input nicht vorhanden";
if(defined($output)) {
    open OUT, ">$output" or die "Fehler bei Outputdatei $output nicht moeglich";
}
else {
    open OUT, ">out.dat" or die "Fehler bei Outputdatei out.dat";
}

while ( $full = <IN> ) {
    chop $full;
    print OUT "Filename: $full \n";
    print OUT "-----\n";
    open SUCHIN, "<$full" or die "Suchdatei nicht gefunden";
    $counter = 0;
    while ($full = <SUCHIN>) {
        chop $full;
        print $full . "\n";
        $erg = grep(/\b$ele\b/i,$full);
        if($erg != 0) {
            ($col1,$col2,$col3) = split(/[ \t]/,$full);
            if(length($col3) == length($ele)) {
                $counter = $counter + 1;
                print OUT "$full $counter\n";
            }
        }
    }
}

$gcounter = $gcounter + $counter;

```

```

    print OUT "Momentan: $gcounter\n";
    close(SUCHIN);

}

print OUT "Gesamt : $gcounter\n";

close(IN);
close(OUT);

sub usage()
{
    print "checkelege.pl <Suchelement> <Inputdatei> <Outputdatei>\n";
    exit;
}

```

6.2.3 Perl-Tool changepath.pl

Zur Erinnerung werden Dateien bei der Datenausgabe automatisch die Extension "sd" besitzen, damit das Programm "sgplay" ohne große Änderungen sofort auf diese Dateien angewandt werden können. Unter Umständen muß der Dateipfad angepaßt werden. Mit Hilfe von changepath.pl kann dies vorgenommen werden.

```

#!/usr/local/bin/perl
use File::Basename;
local($apname);
local($npname);
local($input);
local($temp);

($input,$apname,$npname) = @ARGV;

(defined $input && defined $apname && defined $npname) || usage();

open IN, "<$input" or die " Inputdatei $input nicht vorhanden";
while ($full = <IN>) {
    chop $full;
    $temp = $full;

    $temp =~ s/$apname/$npname/g;

    print $temp . "\n";
}

sub usage() {
    print "changepath.pl Inputdatei APfad NPfad\n";
    exit;
}

```

6.2.4 Zusätzliche Perl-Tools

In diesem Abschnitt werden zwei kleine Perl-Programme dargestellt, die in der Testphase nicht verwendet werden, dennoch nützlich sein können.

mgrep.pl

Mit Hilfe von "mgrep.pl" kann innerhalb einer Datei nach einem bestimmten Wort gesucht werden.

Usage: mgrep.pl gesuchtes Wort Inputfile

filegrep.pl

Mit Hilfe von filegrep.pl" kann unter Angabe einer Musterdatei und der Inputdatei nach den in der Musterdatei angegebenen Mustern gesucht werden. Die Inputdatei kann wiederum eine Reihe von Dateien enthalten, in denen gesucht werden soll.

Usage: filegrep.pl -patlist Musterdatei -filelist Inputdatei

6.2.5 In- und Ouputdateien

Perl-Schnittstelle

In diesem Abschnitt werden alle Dateien aufgelistet, die beim Testen der Perl-Schnittstelle erstellt und/oder verwendet werden:

- Inputdateien
 - phones.dat
 - words.dat
 - text.dat
 - kombi.dat
 - tags.dat
 - syllables.dat
- Outputdateien
 - phones.out, phones_1.out, phones_2.out
 - words.out, words_1.out, words_2.out
 - text.out
 - kombi.out, kombi_1.out, kombi_2.out
 - tags.out, tags_1.out, tags_2.out
 - syllables.out, syllables_1.out, syllables_2.out
- Vergleichdateien
 - phonesver.dat, phonesver_1.dat, phonesver_2.dat, phones_t.dat, phonesref.out
 - wordsver.dat, wordsver_1.dat, wordsver_2.dat, words_durch.dat, wordsref.out
 - textver.dat, text_stern.dat
 - kombiver.dat, kombiver_1.dat, kombi_2.dat, kombi_t.dat, kombiref.out
 - tagsver.dat, tagsver_1.dat, tagsver_2.dat, tags_appr.dat, tagsref.out
 - syllablesver.dat, syllablesver_1.dat, syllablesver_2.dat, syllablesref.out, syllables_n.dat

Verwendete Dateien bei der Datenbankpopulation

Dieser Abschnitt gibt einen Überblick über all die Dateien, mit denen die Testdatenbank "korpo" mit Daten aus den Verzeichnissen `bf /graugans/cd/stern1` und `/graugans/cd/stern2` ausgefüllt wird (siehe 5.1.2), sowie über diejenige Dateien, die bei der Erstellung einer leeren Datenbank herangezogen werden.

- `phones.in`, `phones2.in`
- `words.in`, `words2.in`
- `sd.in`, `sd2.in`
- `tags.in`, `tags2.in`
- `texte.in`, `texte2.in`
- `syl.in`, `syl2.in`
- `syllables.in`, `syllables2.in`
- `html.in`, `html2.in`
- `korpodb`
- `signaltemplate.dmp`

6.3 Perl-Tools zur Erstellung der Inputdatei im C-Programm

In den vorangehenden Abschnitten wurde viel über die Inputdatei gesprochen, die beispielsweise für den Einlesevorgang in Batch-Form oder für die Formatumwandlung benötigt wird und ein bestimmtes Format vorweisen muß. Solch eine Inputdatei kann man mit einem gängigen Editor erstellen. Es ist aber auf jeden Fall erheblich bequemer, wenn man sich der Hilfe der Tools bedient.

Die in Frage kommenden Tools bestehen im wesentlichen aus zwei kleinen Perl-Programmen: "addpath.pl" und "2msql.pl".

Im folgenden wird das Programm `addpath.pl` als Sourcecode aufgelistet:

```
#!/usr/local/bin/perl
# file: stern2msql; author: wok; date: 7.1.98;
# system: perl5.003; latest revision: wok; 7.1.98
# 1. Argument Inputfile
# 2. Argument Pfad
use File::Basename;
local($pname);
local($input);
($input, $pname) = @ARGV;
(defined $input && defined $pname) || usage();
open IN, "<$input" or die "Inputdatei $input nicht vorhanden";
while ( $full = <IN> ) { chop $full;
    $full = $pname . $full;
    print $full . "\n";
}
```

```

sub usage() {
print "addpath.pl Inputdatei Pfad\n";
exit;
}

```

Und folgende Listing zeigt das Programm 2msql.pl:

```

#!/usr/local/bin/perl
# file: stern2msql; author: wok; date: 7.1.98;
# system: perl5.003; latest revision: wok; 7.1.98
use File::Basename;
print "#! PATH NAME PRE DATE TIME EXT\n";    $--> hier "andern$
while ( $full = <> ) { chop $full;
    ( $name, $path, $suf ) = fileparse( $full, '\.[^\.]+' );
    $name =~ /(.*)(\d\d)(\d\d)(\d\d)([\.\d+]*)/;
    $pre = $1 ? $1 : "\"\"";
    $date = "$4.$3.$2";
    #$time = $5 ? $5 : "\"\"";
    $temp = $5 ? $5 : "\"\"";
    if($temp ne "\"\"") {
        $temp =~ /(\.)(.*)/;
        $time = $2;
    }
    else
    {
        $time = $temp;
    }
    $ext = $suf; $ext =~ s/\.(.*)/$1/;
    print "$path\t$name$suf\t$pre\t$date\t$time\t$ext\n";    $--> hier "andern$
}

```

Die Erstellung der Inputdatei erfolgt in drei Schritten unter Verwendung des Unix-Befels "ls" und des Pipe-Mechanismus nach folgendem Schema:

- Ausgehend vom Verzeichnis, wo die Sprachsignaldateien liegen, wird per `ls -l`-Befehl mit der Option "-l" die Liste der Sprachsignaldateien in eine Datei zwischen gespeichert.
Beispiel: `ls -l /graugans/cd/stern1/phones > temp1.dat`
- Die Datei `temp1.dat` enthält alle Sprachsignaldateien unter dem Verzeichnis `/graugans/cd/stern1/phones`, die in einem einspaltigen Format Zeile für Zeile angeordnet sind. Es fehlt jedoch der benötigte Dateipfad in der Liste. An diesem Punkt wird das Programm `addpath.pl` angesetzt. Es liest jede Sprachsignaldatei aus der zuvor erstellten Datei ein (im Beispiel `temp1.dat`), ergänzt dabei jeweils den Dateinamen um den Pfad, der als Parameter beim Aufruf des Programms unbedingt angegeben sein muß, und schreibt den nun vollständigen Dateinamen in eine andere Datei. An dieser Stelle ist es wichtig, darauf hinzuweisen, daß die Pfadangabe unbedingt mit einem Schrägstrich (/) abgeschlossen sein muß.
Beispiel: `perl addpath.pl temp1.dat /graugans/cd/stern1/phones/ > temp2.dat.`

Überdies muß man bei der Angabe des Pfades Sorgfalt walten lassen, denn falsche Angabe führt zu Fehlern bei Routinen im C-Verwaltungsprogramm (beispielsweise bei der Formatumwandlung).

- In diesem Schritt kommt das Programm `2mysql.pl` ins Spiel, das die im 2. Schritt erzeugte Datei (im Beispiel `temp2.dat`) als Inputdatei entgegennimmt und die endgültige Inputdatei für die Perl-Schnittstelle erzeugt. Der Aufruf sieht wie folgt aus: `perl 2mysql.pl temp2.dat > temp.out`.

```
Das Ergebnis der Outputdatei temp.out ist nachfolgend dargestellt:\\
#! PATH NAME PRE DATE TIME EXT
/graugans/cd/stern1/phones/ dlf960724.1657.phones dlf 24.07.96 1657 phones
/graugans/cd/stern1/phones/ dlf960731.1657.phones dlf 31.07.96 1657 phones
/graugans/cd/stern1/phones/ dlf960801.1656.phones dlf 01.08.96 1656 phones
/graugans/cd/stern1/phones/ dlf960802.1656.phones dlf 02.08.96 1656 phones
/graugans/cd/stern1/phones/ dlf960803.1656.phones dlf 03.08.96 1656 phones
/graugans/cd/stern1/phones/ dlf960804.1656.phones dlf 04.08.96 1656 phones
/graugans/cd/stern1/phones/ dlf960805.1656.phones dlf 05.08.96 1656 phones
/graugans/cd/stern1/phones/ dlf960806.1656.phones dlf 06.08.96 1656 phones
/graugans/cd/stern1/phones/ dlf960807.1656.phones dlf 07.08.96 1656 phones
/graugans/cd/stern1/phones/ dlf960808.1656.phones dlf 08.08.96 1656 phones
/graugans/cd/stern1/phones/ dlf960809.1656.phones dlf 09.08.96 1656 phones
```

An dieser Stelle ist es angebracht, nochmal ausdrücklich darauf hinzuweisen, daß die Anordnung der Informationen über PATH, NAME, PRE, DATE, TIME und EXT in beliebiger Reihenfolge erfolgen kann. Um dies Rechnung zu tragen, kann das Programm `2mysql.pl` an zwei markierten Stellen wie oben dargestellt geändert werden. Als Beispiel werden folgende Änderungen vorgenommen:

Gewünschtes Format der Inputdatei: NAME, PATH und EXT

- Änderung:

```
print "#! NAME PATH EXT\n";
```

- Änderung:

```
print "$name$suf\t$path\t$ext\n";
```

6.4 API-Funktionen des mSQL-Datenbanksystems Release 1.06

In diesem Abschnitt werden einige elementare, im C-Programm eingesetzte API-Funktionen eingeführt und deren Bedeutung für ein besseres Verständnis des Anwendungsprogramms näher erläutert.

- `int mysqlConnect(char* host)`

Diese Funktion stellt eine Verbindung zum Datenbankserver her und müs immer zuerst ausgeführt werden. Der Parameter *host* dient zur Spezifizierung eines Rechners mit einem mSQL-System, wenn er nicht NULL ist. Ansonsten wird der lokale Rechner spezifiziert.

Es wird ein Socket-Deskriptor ungleich -1 zurückgeliefert, der für die gesamte Sitzung immer benötigt ist. -1 ist das Ergebnis, falls die Verbindung fehlschlägt.

- `int mysqlSelectDB(int sock, char* dbName)`

Datenbank selektiert werden. Dies erledigt diese Funktion. Zwei Parameter sind erforderlich: der von `mysqlConnect` zurückgelieferten Socket-Deskriptor und der Name der ausgewählten Datenbank.

Der Return-Wert `-1` zeigt einen Fehler an. Diese Funktion kann mehrmals aufgerufen werden, jeweils auch mit anderen Datenbanken. Auf diese Weise kann man zwischen mehreren gleichzeitig angebotenen Datenbanken wechseln.

- `int mysqlQuery(int sock, char* query)`

Mit Hilfe dieser Funktion kann eine Anfrage formuliert und an den Datenbankserver gesendet werden. Die Verbindung zum Datenbankserver erfolgt über den per `mysqlConnect` hergestellten Socket-Deskriptor. Die Funktion liefert wie üblich `-1` als Return-Wert im Fehlerfall zurück. Ansonsten werden Ergebnisse in einem internen API-Buffer zwischengespeichert.

- `m_result *mysqlStoreResult()`

Abfrageergebnisse müssen vom Anwendungsprogramm gespeichert werden, bevor weitere Anfragen gestartet werden, oder sie werden vom internen API-Buffer gelöscht. Die Funktion `mysqlStoreResult` speichert die Ergebnisse in einer vordefinierten Struktur namens `m_result` ab und macht somit den Weg frei für weitere Anfragen. Ein Handle auf die Struktur `m_result` wird zurückgeliefert.

- `void mysqlFreeResult(m_result *result)`

Gespeicherte Abfrageergebnisse sollen gelöscht werden, damit sie nicht unnötigerweise Speicherplatz wegnehmen. Dies erledigt die Funktion `mysqlFreeResult`, der ein Handler auf die Speicherstruktur `m_result` übergeben wird. Diese Funktion ist das Pendant der Funktion `mysqlStoreResult`.

- `m_row mysqlFetchRow(m_result *result)`

In einem relationalen Datenbanksystem wie `mSQL` wird in der Regel auf die gespeicherten Daten zeilenweise zugegriffen, die einem Dateneintrag in einer Datenbanktabelle entsprechen. Eine Datenzeile wird mit Hilfe der Funktion `mysqlFetchRow` in der Datenstruktur `m_row` abgelegt. Die Struktur `m_row` ist ein gewöhnliches C-Array, dessen Element die Daten der Tabellenspaten enthalten. Der vordefinierte Wert `NULL` wird zurückgeliefert, falls keine Daten vorliegen.

- `int mysqlNumRows()`

Die Gesamtanzahl der Zeilen von einer Abfrage wird durch diese Funktion ermittelt. Dadurch hat der Programmierer ein Werkzeug zur Hand, um beispielsweise eine Schleife zu bilden, innerhalb derer sukzessiv auf jede einzelne Datenzeile (per `mysqlFetchRow`) zugegriffen wird.

- `m_field *mysqlFetchField(m_result *result)`

Jede Spalte einer Tabelle besitzt u.a. einen Typ (`char`, `reel`, `int`), und einen Namen. Definitionsinformationen von Tabellenspalten werden unter Mitwirkung der Funktion `mysqlFetchField` gewonnen und in der Struktur `m_field` abgelegt.

- `int mysqlNumFields(m_result *result)`

Diese Funktion liefert die Gesamtanzahl der Tabellenspalten, die in einer Anfrage spezifiziert sind.

- `m_result *mysqlListDBs(int sock)`

Wie bereits erwähnt wird eine Verbindung zum Datenbankserver durch einen Socket-Deskriptor repräsentiert. Mit diesem Socket-Deskriptor als Argument erhält man per `mysqlListDBs` Namen der Datenbanken, die dem Datenbankserver bekannt sind (angebunden sind).

- `m_result *mysqlListFields(int sock, char* tablename)`

Im Gegensatz zur Funktion `mysqlFetchField` wird die Funktion `mysqlListFields` auf eine Tabelle angewandt, die dann Definitionsinformationen über die betroffene Tabelle liefert.

- `int mysqlClose()`

Die Funktion beendet ordnungsgemäß die Verbindung zum Datenbankserver. Es ist dabei notwendig, daß derselbe Socket-Deskriptor als Argument verwendet wird, der zuvor von der Funktion `mysqlConnect` geliefert wurde.

Kapitel 7

Installation, Konfiguration und Verzeichnisstruktur

Dieses Kapitel behandelt folgende Themen:

- Installation von mSQL
- Kompilierung und Installation vom C-Programm
- Verzeichnisse, in denen die benötigten Dateien vorliegen

7.1 Installation des Datenbanksystems mSQL

Die Installation des Datenbanksystems mSQL ist nicht kompliziert und besteht aus folgenden Schritten:

- Als erstes soll das Datenbanksystem mSQL von einem geeigneten Server auf die lokale Festplatte kopiert werden, sofern es noch nicht vorliegt. Die http-Adresse `www.hughes.com.au` wäre die erste Adresse. Die zu kopierende Datei heißt in diesem Fall `"msql-1.0.16.tar.gz"`.
- Da mSQL in der Regel in gepackter Form vorliegt, soll es als nächstes ausgepackt werden (mit `gunzip` und `tar -xvf`).
- Mit dem Befehl `tar -xvf` wird die Verzeichnisstruktur des mSQL-Pakets mit ausgepackt. Es wird nun in das Hauptverzeichnis dieser Verzeichnisstruktur namens `"msql-1.0.16"` gewechselt und anschließend der Befehl `make target` eingegeben.
- In Abhängigkeit von dem verwendeten Betriebssystem wird ein Zielverzeichnis angelegt. Bei IRIX 6.2 lautet dieses Verzeichnis `"targets/IRIX-6.2.IP20"`. Es wird in dieses Verzeichnis gewechselt. Dann gibt man den Befehl `setup` ein. Wenn `"setup"` ausgeführt wird, sind folgende Angaben erforderlich:
 - Installationsverzeichnis
 - Ausführungsrecht
 - Benutzername
 - Verzeichnis für die PID-Datei

Dabei ist zu beachten, daß das angegebene Verzeichnis für die PID-Datei bereits angelegt ist. Die dabei getätigten Angaben werden in der Datei `site.mm` gespeichert.

- Am Anschluß daran wird der Befehl *make install* gestartet. Als Voreinstellung wird das Datenbanksystem mSQL in das Verzeichnis `/usr/local/Minerva` installiert. Da nicht jeder Schreibzugriff auf das Verzeichnis `/usr/local` hat, müssen unter Umständen in der Datei *site.mm* einige Änderungen gemacht werden. Dies betrifft in erster Linie folgende Einstellungen:
 - `ROOT="Username"`
 - `INST_DIR="Installationsverzeichnis"`
 - `PID_FILE="Verzeichnis für Pid-File"` (in der Regel gleich dem Installationsverzeichnis)
 - `CC=cc` (Einstellung des verwendeten Compilers, GNU-Compiler `gcc` funktioniert nicht)
- Wenn der Befehl `"make install"` keine Fehlermeldung anzeigt, dann ist der Installationsvorgang erfolgreich beendet. Es werden zusätzlich zwei Unterverzeichnisse angelegt, nämlich `"include"` und `"lib"`. Das Verzeichnis `include` enthält u.a. die Include-Datei `mysql.h`, die bei der Kompilierung des C-Verwaltungsprogramms verwendet wird. Das Verzeichnis `lib` enthält die betriebssystemabhängige Datei `libmysql.a`. Man kann von nun an mit mSQL arbeiten.

7.2 Verzeichnisstruktur

Im Rahmen der Programmentwicklung sowie während der Testphase entsteht eine Vielzahl von Dateien, die strukturiert auf der Festplatte abgelegt werden sollen. Somit bekommt man einen Überblick über alle relevanten Dateien. Daraus ergibt sich eine Verzeichnisstruktur, die nachfolgend beschrieben wird.

- *Verzeichnis verwalt*
Dieses Verzeichnis enthält die komprimierte Tar-Datei `"verwalt.tar"`, die u.a. alle C-Module, Include-Dateien, Library-Dateien und die Makefile zum Kompilieren enthält.
- *Verzeichnis analyse*
Dieses Verzeichnis enthält das Perl-Programm `analyse.pl` zur Datenauswertung und Datenausgabe.
- *Verzeichnis Hilfsprogramm*
In diesem Verzeichnis sind die benötigten Hilfsprogramme wie z.B. `addpath.pl`, `2mysql.pl`, `checkle.pl` ... abgelegt.
- *Verzeichnis doc*
Dieses Verzeichnis enthält die Ausarbeitung der Diplomarbeit in Tex-Format und Postscript-Format und die Datei `README`, in der kurz und bündig beschrieben wird, wie das Datenbanksystem mSQL installiert wird.
- *Verzeichnis dbdump*
Dieses Verzeichnis enthält die Backupdatei der Datenbank `korpora`, welche mit Hilfe des Tools `msqldump` erstellt wurde.
- *Verzeichnis input*
In diesem Verzeichnis sind alle Input-Dateien zu finden, die beim Einlesen von Daten in die Tabelle `FILE` verwendet werden. Diese Input-Dateien stellen im Grunde genommen die Dateilisten in den Verzeichnissen `/graugans/cd/stern1` bis `/graugans/cd/stern6` dar.
- *Verzeichnis output*
In diesem Verzeichnis sind Dateilisten abgelegt, die bei der Datenauswertung durch das Perl-Programm erstellt werden.

- *Verzeichnis `mysqlsource`*

Dieses Verzeichnis enthält das komprimierte mSQL-Softwarepaket `mysql-1_0_16.tar.gz`.

Dabei ist es darauf hinzuweisen, daß es zusätzlich ein komprimiertes Dateipaket namens **`korporadb.tar.gz`** gibt, die alle oben erwähnten Dateien mit der oben dargestellten Verzeichnisstruktur enthält. Mit diesem Paket ist es nicht schwierig, die gesamte Datenbanksoftware auf einen anderen Rechner zu übertragen.

7.3 Kompilierung und Installation vom C-Programm

Dieser Abschnitt widmet sich auf ausführliche Weise der Kompilierung und der Installation des C-Verwaltungsprogramms. Folgende Schritte sind durchzuführen:

- Die komprimierte Datei `verwalt.tar` auspacken
- Das C-Programm mit dem Befehl "make" kompilieren.

An dieser Stelle ist es erwähnenswert, darauf hinzuweisen, daß die Include-File `mysql.h` und die Library `libmysql.a` bei der Kompilierung benötigt sind. Diese zwei Dateien stammen von der mSQL-Softwarepaket. Es ist offensichtlich verständlich, daß die mSQL-Software vorher installiert werden muß, denn die Library `libmysql.a` ist betriebssystemabhängig und wird bei der Installation des Datenbanksystems mSQL erzeugt.

Nach der Installation des C-Verwaltungsprogramms soll anschließend die Einstellung der für den Betrieb notwendigen Umgebungsvariablen vorgenommen werden. Dabei handelt es sich um die zwei Umgebungsvariablen "ExPho" und "ALIGNERBIN". Dies geschieht durch entsprechende Einträge in der Datei `.cshrc` wie folgt:

- Einstellung von ExPho:
`setenv ExPho /usr/localtmp/ExPhoDB` (siehe auch 3.2.3).
- Einstellung von ALIGNERBIN:
`setenv ALIGNERBIN /usr/local/Aligner/bin`

Dabei wird zum einen vorausgesetzt, daß das Verzeichnis `/usr/local/tmp/ExPhoDB` vorhanden ist. Dieses Verzeichnis ist jedoch frei wählbar. Wichtig ist nur, daß die Umgebungsvariable `ExPho` einen Wert besitzt. Zum anderen wird der Umgebungsvariablen `ALIGNERBIN` der Wert `/usr/local/Aligner/bin` zugewiesen, weil die Transformationsroutinen in diesem Verzeichnis vorliegen. Sollten sie woanders installiert werden, muß die Umgebungsvariable `ALIGNERBIN` angepaßt werden.

Kapitel 8

Zusammenfassung und Aussicht

Auf der Grundlage der projektspezifischen Aufgabenstellung wird im Rahmen dieser Diplomarbeit ein Prototyp entwickelt, der einerseits dem Anforderungsprofil weitestgehend zu genügen und andererseits die funktionelle Umsetzung verständlich zu demonstrieren versucht. Dies wird zum einen ermöglicht durch die modulare Aufteilung zwischen einem C-Verwaltungsprogramm und einer Perl- Auswertungsschnittstelle. Zum andern ist die Bedienung des C-Verwaltungsprogramms sowohl menügesteuert als auch per Kommandozeileoptionen bedienbar, so daß aus Benutzersicht die Bedienung insgesamt flexibel gehalten wird. Das Datenbankmodell ist so strukturiert, daß es genügend Raum läßt für anderweitige Erweiterungen. Überdies wird dabei versucht, die von der eingesetzten mSQL-Version unterstützten Funktionalitäten so weit wie möglich auszuschöpfen.

Trotzallerdem ist und bleibt das aus dem Analyse- und Verwaltungsprogramm bestehende Programmkonstrukt ein Prototyp. Dies bedeutet nicht anderes als daß weitere Fragen hinsichtlich des Datenbankmodells, des eingesetzten mSQL-Datenbanksystems und der Implementierungsmethodik (Verwaltungsprogramm in C, Auswertungsschnittstelle in Perl) noch offen sind. Hier für sind solche Fragen zu beantworten:

- Mit den eingeschränkt unterstützten Features der momentan eingesetzten Version des mSQL-Datenbanksystems kann ein Prototyp gemäß der Zielsetzung erstellt werden. Wenn der Funktionsumfang jedoch erweitert werden soll, muß eine neue mit mehr Funktionalitäten ausgestattete Version von mSQL eingesetzt werden. In diesem Zusammenhang stellen sich nun folgende Fragen: Soll eine neue leistungsfähigere Version von mSQL eingesetzt werden? Wenn Ja, welche UNIX-Plattform käme dann in Frage?. Wird auch die aktuell verwendete UNIX-Plattform SGI-ULTRIX bei der neuen Version auch unterstützt?
- Sollen Zugriffsrechte auf Sprachsignaldateien in Zukunft zusätzlich innerhalb des Verwaltungsprogramms geregelt werden? Beispielsweise kann die Formatumandlung nur auf diejenige Sprachsignaldateien angewandt werden, welche Schreib/Lese-Zugriffsrecht besitzen
- Soll das Verwaltungsprogramm in eine andere, intepretierbare Implementierungssprache auf Kosten der Geschwindigkeit konvertiert werden? (etwa msqIPerl)

Dies hat zur Folge, daß Erweiterungs- und/oder Änderungsbedarf in Zukunft eventuell entstehen würde.

Eine gewisse Flexibilität zeichnet die Perl-Schnittstele aus, die ja in Perl programmiert ist. Überdies bietet dieses Perl-Programm ein flexibles Kommando-Interface an, so daß potentielle Erweiterungen im Bereich der Ergebnisauswertung sowie der Ergebnisanalyse in diese Richtung gehen sollen

Obendrein ist es hinsichtlich der Programmführung benutzerfreundlicher, wenn das C-Verwaltungsprogramm mit einer Benutzeroberfläche versehen ist und die Menüpunkte somit mit der Maus bedient werden können. Dafür wäre das `msqltcl`-Paket (Tk/Tcl) geeignet, das jedoch wahrscheinlich nur mit einer höheren Version von `mSQL` zusammenarbeiten kann. Dazu schließt sich der Kreis – eine neue `mSQL`-Version ist für die eventuellen Erweiterungen unbedingt notwendig.

Kapitel 9

Literatur

1. <http://www.hughes.com.au>
2. <http://www.blnet.com/msqlpc>
3. <http://www.stonehenge.com/merylin>
4. <http://www.swl.fh/heilbronn.de/msql>
5. <http://www.est.fn.bw.schule/kurse/mSQL/index.html>
6. <http://www.perl.com>
7. <http://www.perl.guru.org>
8. <http://www.perl.net>
9. <http://www.progsources.com/perl.html>
10. <http://www.sol.mmtlc.utep.edu/msql.html>
11. <http://www.leo.org/pub/comp/msql>
12. <http://c4.hrz.uni-giessen.de/hrz/software/msql/homepage.html>
13. <http://mail.erste.de/docs/pg95/doc2/>
14. http://www.cglis.com/msql_keeper
15. <http://www.oase-shareware.org/shell/categories.html>
16. <http://www.activestate.com/reference/documentation.htm>
17. <http://www.mcp.com/personal>
18. <http://ipo53.informatik.htw-dresden.de/JAVA11/MySql>
19. <http://www.omcnet.de/support/mysql-toc.html>
20. <http://www.postgresql.org/index.html>
21. <http://sugar.komm.hdk-berlin.de/pgsql/>
22. <http://tina.ind.ba-heidenheim.de/dokumentation/postgresql/programmer>
23. Wall, Larry; Christiansen, Tom; Schwartz, Randal L.: Programming Perl

24. Hall, Joseph N.; Schwartz Randal L.: Effective Perl Programming. Addison/Wesley Longman Amsterdam
25. Edward S. Peschko, Michele DeWolfe: Perl 5 Complete. McGraw-Hill
26. Röhrig, Bernhard, Datenbanken mit Linux, Vater Stetten: C&L, 1998
27. Brian Jepson, David J Hughes: Official Guide to MiniSQL 2.0, John Willey & Sons
28. Kemper, Alfons: Datenbanksysteme: eine Einführung. Oldenbourg, 1977
29. Beger, Peter: Datenbankabfrage mit SQL. Mark und Technik, 1989
30. Moos, Alfred: Datenbank-Engineering: Analyse, Entwurf und Implementierung relationaler Datenbank mit SQL. Vieweg, 1997
31. Moss, Alfred, Daues Gerhard: SQL-Datenbanken: Der Weg vom Konzept zur Realisierung. Vieweg, 1991
32. Kohler, Klaus J.: Einführung in die Phonetik des Deutschen - 2., neubearb. Aufl. - Berlin:Schmidt, 1995
33. Neppert, Joachim: Magnus Petursson: Elemente einer akustischen Phonetik:
34. Drosdowski, Günther; Eisenberg, Peter: Duden "Grammatik der deutschen Gegenwartssprache-Mannheim: Dudenverl., 1995 - Hamburg: Buske, 1992
35. Rapp, S.: Automatic phonemic transcription and linguistic annotation from known text with Hidden Markow Models. An Aligner for German. Proceedings of ELSNET goes east and IMACS workshop "Integration of Language and Speech in Academy and Industry", Moscow, 1995
36. <http://www.ims.uni-stuttgart.de>