ORACLE und die Analyse des Äußerungsaufbaus

Rainer Dietrich, Heidi Egel,
Bärbel Maier-Schicht, Marion Neubauer

Bericht Nr. 32
Februar 1991

Arbeiten aus dem Sonderforschungsbereich 245
"Sprechen und Sprachverstehen im sozialen Kontext"
Heidelberg/Mannheim

Kontaktadresse:

Universität Heidelberg
Institut für Deutsch als
Fremdsprachenphilologie
Plöck 55, 6900 Heidelberg

Diese Arbeit ist im Sonderforschungsbereich 245 der Universitäten Heidelberg und Mannheim enstanden sowie dem Institut für deutsche Sprache, Mannheim. Wir danken der Deutschen Forschungsgemeinschaft für die Förderung unserer Arbeit.

ISSN 0937-6224

INHALTSVERZEICHNIS

Zusammenfassung

Berichtet wird über die Konzeptionalisierung, Anwendungsentwicklung und Implementierung einer Datenbankanwendung für das Forschungsprojekt "Äußerungsaufbau" im Sonderforschungsbereich 245 "Sprechen und Sprachverstehen im sozialen Kontext" Heidelberg/Mannheim. Bei dem verwendeten Datenbankmanagementsystem handelt es sich um das Programmpaket ORACLE.

In den Abschnitten 1 bis 3 werden die Projektfragestellung, zu deren Klärung die Datenbank beitragen soll sowie die Notwendigkeit einer Datenbank für dieses Vorhaben dargelegt. Im Abschnitt 4 wird begründet, weshalb ORACLE als Datenbankmanagementsystem verwendet wird. Die Abschnitte 5 bis 7 sind mit der Aufbereitung der Projektdaten für die Datenbank, den Möglichkeiten, die ORACLE für das Arbeiten mit den Daten bietet, der Anwendungsentwicklung und Implementierung befaßt. Abschnitt 8 ist "zu Nutz und Frommen" all jener bestimmt, die sich auf ein ähnliches Vorhaben einlassen wollen.

Abstract

We describe the development of a database application to be used by the linguistic research project "Äußerungsaufbau" which is one of several research projects of the "Sonderforschungsbereich 245 "Sprechen und Sprachverstehen in sozialen Kontext" Heidelberg/Mannheim. The software that has been used is ORACLE, a relational database management system.

The first three parts deal with research questions and ways in which a database might contribute to their solution. Part 4 is a brief discussion why ORACLE has been used. Parts 5 through 7 give a detailed description of the data, how they have been pepared, and how the application has been developed and implemented. The last part tries to give some hints to all those who will be in a comparable situation.

ORACLE und die Analyse des Äußerungsaufbaus

(Eine sprachwissenschaftliche Datenbankanwendung)

0. Inhalt und Zweck dieses Papiers

Bei einem Menschen mit (Resten von) humanistischem Bildungshintergrund könnte der Titel dieses Heftes Assoziationen in Richtung auf die griechische Mythologie auslösen, und damit wäre genau eines von den Mißverständnissen geschehen, die zu vermeiden zu den Zielen dieser kleinen Schrift zählt. Sie handelt nicht von der Interpretation delphischer Rätselsprüche, wie vielleicht der erste Teil des Titels nahelegt, sondern von der Verwendung eines Programmpakets namens ORACLE[1)] mit dem Ziel, größere Mengen von sprachlichen Äußerungen nach einer größeren Anzahl von Kategorien im Zusammenhang leichter, schneller, sicherer, umfassender und genauer linguistisch zu analysieren. Dargestellt werden Ausgangslage, Vorbereitung, Planung und Realisierung einer ORACLE-Anwendung in einem Projekt zur Analyse von pragmatischen und inhaltlichen Prinzipien der Wortstellung in Äußerungen gesprochener Sprache. Die Fragen, auf die wir[2)] dabei gestoßen sind, die Erfahrungen, die wir dabei gemacht haben und die Lösungen, die wir (einstweilen) erarbeitet haben, sind nicht alle auf beliebige andere Fragestellungen übertragbar und damit nicht alle gleichermaßen des Mitteilens wert. Hier werden natürlich in erster Linie die übertragbaren dargestellt. Dabei kann es das Erfordernis der Verstehbarkeit nötig machen, das eine oder andere projektspezifische Detail einzuflechten; die entsprechenden Stellen lassen sich leicht erkennen; es wurde aber auch da versucht, die allgemeinere Relevanz solcher Besonderheiten durch Hinweise auf ähnlich gelagerte Problematiken anderen Inhalts zu erhöhen. Am wenigsten gelungen ist dies, wo die Lösungen stark durch die lokalen und projektspezifischen Hardware-Vorgaben bestimmt waren, wie in den Abschnitten 5, 6 und 7 dargestellt wird. Dennoch mag das Heft also einerseits Sprachwissenschaftlern etwas sagen, die mit großen Belegzahlen und vielen Deskriptoren arbeiten, anderseits DV-Experten, die sich für Anwendungen des relationalen Datenbankmodells in traditionell eher DV-ferneren Gebieten interessieren. Der Aufbau der Darstellung ergibt sich aus der Sache. Natürlich liegt es nahe, zunächst mit dem Projekt selbst bekanntzumachen, dem die ORACLE-Anwendung von Nutzen sein soll, und zu zeigen, welche besonderen Erfordernisse

---

1) Im Unterschied zu so transparenten Akronymen wie AQUAD, LISP, FORTRAN ist ORACLE offenbar keine Abkürzung sondern tatsächlich ideenverwandt mit dem weltlichen Mitteilungsorgan der griechischen Götterwelt. Die Experten, auf die wir uns in dieser Mitteilung beziehen können, Craig W. Slinkman (B995CWS@UTARLVM1.BITNET), und Toni DiPesa (DIPESA@WOODY.MIT.EDU.BITNET), räumen allerdings auch ein, daß die Ideenverwandtschaft so lose ist, wie es eben nach rund zweieinhalbtausend Jahren nicht ausbleibt. Es sei ihnen an dieser Stelle herzlich für ihre Auskünfte gedankt.

2) Die Arbeit über die hier berichtet wird, war eine ausgesprochene Gemeinschaftsarbeit einer Gruppe von Mitgliedern des SFB 245 "Sprechen und Sprachverstehen im sozialen Kontext" der Universitäten Heidelberg und Mannheim. Die Hauptarbeit und die Erstellung dieses Berichts lagen in den Händen von Heidi Egel und Marion Neubauer sowie der anderen im Inhaltsverzeichnis einzeln genannten Verfasser. Zeitweise, aber nicht minder wesentlich beteiligt waren Katharina Bremer, Mary Carroll, Ingeborg Gutfleisch, Uli Scharnhorst, Richard Siebrecht und vor allem Stefan Buchta.

den hier beschriebenen Einsatz der Datenverarbeitung begründet haben. Das wird in den ersten drei Abschnitten dargestellt. Dann muß ein grobes Bild des relationalen Datenmodells und des Programmpakets ORACLE vorliegen; davon handelt Abschnitt 4. Die Einzelheiten werden dann in den Abschnitten 5 bis 7 beschrieben, die den Hauptteil des Hefts ausmachen. Am Ende werden in einer Reihe von Merkpunkten die Erfahrungen resümiert, die wir besonders denjenigen ans Herz legen, die vor ähnlichen Vorhaben stehen. Wer sich nicht schlüssig ist, ob der Inhalt dieses Papiers ihm etwas geben kann, sollte die Lektüre also vielleicht mit Abschnitt 8 beginnen.

1. Gegenstand und Fragestellung des Projekts "Äußerungsaufbau"

Die leitende Fragestellung im Projekt "Äußerungsaufbau in elementaren und ausgebauten Sprachen" richtet sich auf die lineare Folge von Wortformen in spontaner mündlicher Rede und gilt den semantischen und pragmatischen Regularitäten, die ihre Reihenfolgen in Äußerungen in wechselnden Kontexten regeln. Eine Äußerung wird als die Verbalisierung (einer Vorstellung) eines Sachverhalts, d.h. einer Konstellation aus Personen/Objekten, zeitlichen und räumlichen Bestimmungen und eines Geschehens oder Zustands und dessen Modalität aufgefaßt. Welche lexikalischen Mittel und welche syntaktische Gliederung je Sachverhalt und Äußerung vom Sprecher gewählt werden, bestimmt sich bekanntlich nach einer Vielzahl von Voraussetzungen: dem Sachverhalt selbst, dem in der Sprache vorhandenen Repertoire, dem individuellen Repertoire des Sprechers, dem, was zuvor dazu gesagt worden ist und nach den situativen Gegebenheiten. Das Projekt als Ganzes untersucht die genannten Regularitäten zunächst im Deutschen und im Englischen sowie im zweitsprachlichen Deutsch und Englisch italienischer Erwachsener; ergänzend und zur punktuellen Kontrolle werden auch Betrachtungen zum Italienischen selbst und zum Lernerdeutsch chinesischer Erwachsener angestellt. Die Analysen setzen dabei an dem Punkt ein, an dem - überzeichnend einfach gesprochen - der Sprecher sich der kommunikativen Aufgabe klar ist, die er mit dem aktuellen Text beantworten will, in Bearbeitung dieser Aufgabe eine gedankliche Struktur, jedenfalls ein Stück weit entwickelt oder auch schon verbalisiert hat und nun darangeht, eine weitere Proposition zu erstellen, und die Äußerung, mit der er sie ausdrücken wird, dem Text hinzufügt. Wir betrachten die Äußerung gewissemaßen in dem Zustand, in dem die lexikalische Auswahl zur Beschreibung des in der Äußerung auszudrückenden propositionalen Inhalts abgeschlossen ist und die strukturelle Gliederung sozusagen feststeht[3]. Das Ziel ist zu ermitteln, nach welchen Prinzipien die lineare Ordnung der Segmente sich bestimmt. In Frage kommen die Gliederung der Information im bisherigen Textaufbau, beim Textanfang der Bezug zur kommunikativen Aufgabe oder Zielsetzung direkt, und zum zweiten der Inhalt der Äußerung selbst, deren Reihung Gegenstand der Analyse ist. Wir gehen also davon aus, daß schon auf der Ebene der präverbalen propositionalen Strukturierung die Information in eine Reihenfolge gebracht wird, sei es als ganze Struktur oder auch "portionsweise", und daß sich danach die Wahl der geeigneten sprachlichen Mittel orientiert, mit denen der

3) In Herrmann (1985, S. 257) findet sich zu diesem Teilgebiet der Sprachtheorie als Einleitung zu den von ihm angebotenen Strukturierungen eine Feststellung, die bis heute wenig von ihrer Gültigkeit und nichts von ihrer Schubkraft verloren hat:"Der Encodier-Mechanismus mutet einstweilen als eine hermetische und geheimnisvolle "Blackbox" an."

Proposition sprachlich Ausdruck verliehen wird.[4]. Der verbleibende Spielraum ist auch bei nicht übermäßig langen Äußerungen noch so reich, daß er, soweit zu sehen ist, noch für keine Sprache in einer überschaubaren Weise geordnet worden ist. Eine überzeugende Illustration der Vielfalt von Möglichkeiten gibt Altmann (1987). Die Frage, welche Faktoren aber hinter dieser Variation stehen, d.h. welche Bedeutungs- und welche Äußerungsbedingungen im Zusammenspiel für eine gegebene Proposition zu der schließlich produzierten Reihenfolge führen, ist immer noch wenig geklärt, was sicher auch damit zu tun hat, daß dieses Zusammenspiel je Sprache verschieden ist. Was man bisher kennt, sind einige wenige isolierte Prinzipien in einzelnen Sprachen, deren augenfälligste im Deutschen schon von Behaghel (1932) genannt wurden, und ein theoretischer Entwurf der Struktur der gesamten Satzerzeugung, Levelt (1989). Beide machen die Hauptschwierigkeit augenfällig, auf die weitere Klärungsversuche bisher immer wieder gestoßen sind, nämlich die Komplexität des Problems. Sie folgt aus der anzunehmenden Zahl der beteiligten Faktoren, aus den gegenseitigen Abhängigkeiten und aus den Zusammenhängen zwischen den beteiligten Ausdrucksmitteln: Lexik, Struktur, Reihenfolge und Prosodie. Daß dies alles und sicher noch viel mehr also mit allem zusammenhängt, ist letztlich auch nicht strittig und wird durch feinfühlige Konversationsanalysen immer wieder mal beschrieben. Zu Erkenntnissen über die involvierten Faktoren und ihr Zusammenspiel wird man aber eher kommen, indem man für Zwecke der schrittweisen Analyse die Komplexität zeitweise reduziert und sich auf sinnvoll eingegrenzte Teilbereiche von Zusammenhängen konzentriert. Auf dieser Linie liegt die Arbeit des Projekts "Äußerungsaufbau" und von daher bestimmen sich die grundsätzlichen Vorgehensweisen. Der Gegenstand der Projektarbeit ist, wie gesagt, die Wortstellung, und die Frage richtet sich auf sprachliche und auf außersprachliche Bedingungen, die die Reihenfolge der Elemente in der Äußerung regulieren - und auf ihr Zusammenspiel. Von daher ergibt es sich als sinnvoll, nach den bisher als relevant anzunehmenden Bedingungen systematisch verschiedene Korpora von Texten zusammenzustellen, schriftlich zu dokumentieren, in die für die Analyse relevanten Segmente zu zerlegen und alles nach all den Eigenschaften zu beschreiben, die als reihenfolgesensitiv verdächtig sind. Dann beginnt die eigentliche Arbeit, nämlich herauszufinden, welche Eigenschaft bzw. Äußerungsbedingung welche Wirkung auf die Reihenfolge hat - und in welcher Abhängigkeit von welchen anderen Eigenschaften oder Bedingungen. Trifft die Analyse auf Fragen, die mit den vorhandenen Korpora nicht zu beantworten oder zu überprüfen sind, muß die Datenbasis erweitert werden, und die neuen Daten müssen auch den angedeuteten Schritten unterzogen werden. Welche Kategorien und welche Segmente im Projekt im einzelnen behandelt werden, wird im Abschnitt 5.1 veranschaulicht.

---

4) Ein differenziertes Bild von dem Zusammenwirken von syntaktischen und konzeptuellen Eigenschaften und kontextuellen Bezügen der Einheiten in einer zu linearisierenden Struktur präsentiert Levelt (1989, S. 236-246).

2. Warum eine Datenbank?

Die Frage, ob für die Arbeit in einem Projekt wie dem eingangs skizzierten oder einem mit ähnlichem Komplexitätsgrad und vergleichbarer Zielsetzung eine Datenbank auf einem Rechner aufgebaut werden soll, wird nach allen bisherigen Erfahrungen im Zuge der fortschreitenden Arbeit mehrmals virulent. Taucht die Frage zum ersten Mal auf, hoffentlich in einer frühen Phase des Projekts, ist die Antwort im allgemeinen klar positiv oder negativ. Zum letzten Mal stellt sich die Frage dann am Ende der Projektarbeit, und auch dann ist die Antwort meist klar; nicht so klar ist hingegen, ob es noch dieselbe ist wie am Anfang. Wichtig ist jedenfalls der relative Zeitpunkt, zu dem man sich mit der Frage beschäftigt. In der Vorrede zur ersten Auflage seines "Wörterbuchs der Althochdeutschen Sprache" von 1834 finden sich folgende bewegenden Worte des Verfassers (Eberhard Graff): "Welche lange mühselige Arbeit hat dieses Werk mir auferlegt, welchem Gram und Kummer mich ausgesetzt, welche Opfer von mir gefordert ! (...) Nur durch frommes, vertrauensvolles Gebet und durch treuen, unermüdlichen Fleiß bin ich - wenn auch spät - erst beim Sinken meines Lebens, halb erblindet und an Geist und Körper geschwächt, der Vollendung meines Werkes nahegekommen." Da neigt man rasch zu der Annahme, jemand wie Graff wäre heute gut beraten, wenn er neben Gebet und Fleiß auch noch einen leistungsfähigen Rechner und eine komfortable Datenbanksoftware einsetzen würde. Eigentümlicherweise geschieht das aber auch heute noch in den wenigsten lexikographischen Großprojekten - zumindest an Deutschlands Universitäten. Der Grund dafür ist, daß die Frage danach zu spät gestellt worden ist. Stehen die ersten geordneten und eingeordneten Erkenntnisse erst einmal auf fünfzig- bis sechzigtausend Karteikärtchen, dann bedeutet die Überführung all dieser Informationen in eine Datenbank entweder eine ernsthafte Verzögerung oder den Einsatz von zusätzlichen Mitteln für Geräte, Experten und Arbeitszeit. Beides stößt auf ernsten und berechtigten Widerstand der Projektförderer. Und so bleibt es dann im allgemeinen bei den bis dahin eingespielten Mitteln und Verfahren. Ohne Gebete und Fleiß im geringsten in Zweifel ziehen zu wollen, lehrt uns das Schicksal des Computereinsatzes in den lexikographischen Großprojekten jedenfalls, wie bedeutsam der Zeitpunkt der Entscheidung über Computereinsatz in komplexen Projekten der Sprachbeschreibung ist. Im hier vorliegenden Fall stand die Arbeit den bisherigen Schätzungen nach im ersten Fünftel der Gesamtdauer. Ob schon das unökonomisch spät oder noch gerade rechtzeitig ist, mag man aus den Darstellungen ableiten; wir werden darauf nur kursorisch, im Zusammenhang mit dieser oder jener Einzelheit im Hauptteil explizit eingehen. Jetzt sollen erst einmal die Umstände vorgetragen werden, die zu einer klar positiven Entscheidung geradezu nötigen.

2.1. Komplexität

An welcher Stelle und an welcher Position relativ zu anderen Wörtern ein Wort in einer Äußerung erscheint, ist offenbar nicht beliebig. Das kann man sich leicht vor Augen führen, indem man beispielsweise die Wörter des vorangehenden Satzes ohne Rücksicht auf grammatische Richtigkeit durchpermutiert. Schon wenn man die Positionen der ersten beiden vertauscht, entsteht eine Anordnung, die mit den syntaktischen Regeln des Deutschen nicht verträglich ist. Das heißt aber nicht, daß - umgekehrt - alle grammatisch zuläs-

sigen Reihenfolgen strikt durch syntaktische Regeln bestimmt sind, daß also für eine gegebene Ausdrucksabsicht durch syntaktische Regeln eine und genau eine Linearisierung bestimmt ist. Auch das läßt sich am ersten Satz exemplifizieren. Ohne gegen syntaktische Einschränkungen zu verstoßen, könnte man denselben Inhalt mit denselben Wörtern auch in anderer Reihenfolge ausdrücken, zum Beispiel, indem man den darin enthaltenen Hauptsatz voran- und den Nebensatz hintanstellt: "Offenbar ist nicht beliebig, an welcher Stelle ein Wort erscheint....". Aber auch diese Variation sozusagen innerhalb des syntaktisch Möglichen ist nicht beliebig, sondern regulär. Während aber die syntaktischen Einschränkungen relativ gut bekannt sind, liegen die außersyntaktischen Gesetzmäßigkeiten, nach denen die Variation organisiert ist, noch sehr im Dunkeln. Zahlreiche Detailstudien haben allerdings vielerlei Einzelzusammenhänge aufgezeigt, vornehmlich für die Besetzung der Anfangsposition der Äußerung. Nimmt man alles zusammen, sind an den gesuchten Regularitäten rund fünfundzwanzig Bedingungen unterschiedlicher Arten von Äußerungsteilen und in unterschiedlichen gegenseitigen Abhängigkeiten beteiligt. Darunter sind - im vorliegenden Projekt - so globale wie die, ob die Äußerung dem Englischen, Deutschen oder einer Lernersprache von Englisch- oder Deutschlernenden entstammt. Weitere Bedingungen - z.B. von referentiellen Einheiten auf der Phrasenebene - sind, ob die Sache, die mit der Phrase bezeichnet wird, im laufenden Diskurs schon erwähnt worden ist oder nicht und falls ja, ob dies unmittelbar zuvor oder länger zuvor der Fall war, ob mit der Phrase eine Person, ein Ding, ein Ort, eine Zeit bezeichnet wird, ob ggf. die bezeichnete Person das in der Äußerung bezeichnete Geschehen auslöst, durchführt o.a. usw. Geht man von der durchaus reellen Annahme aus, jede Bedingung weise im Schnitt fünf Ausprägungsmöglichkeiten auf, rund ein Drittel wären Bedingungen auf Diskursebene, also Eigenschaften von Texten als ganzen, rund die Hälfte solche auf Äußerungsebene und ein Sechstel solche auf Phrasenebene, so würde das für die Beschreibung eines Textes im Korpus bedeuten, daß etwa acht Werte auf Diskursebene, rund ein Dutzend pro Äußerung und fünf bis sechs pro Phrase festgehalten werden müssen. Nimmt man weiterhin an, das Korpus des Projekts enthielte einstweilen 40 Texte mit durchschnittlich je 30 Äußerungen zu je fünf Phrasen, so ergäbe das 320 (40 x 8) Werte auf der Diskursebene, 14400 (40 x 30 x 12) auf der Äußerungsebene und mindestens 30000 (40 x 30 x 5 x 5) auf Phrasenebene. Nicht in diesen Bedingungen enthalten ist natürlich die Reihenfolgeinformation, die ja so etwas wie die abhängige Variable darstellt. Die Analyse geht nun natürlich nicht voraussetzungslos sozusagen von Null aus. Das heißt, es müssen nicht alle rechnerisch möglichen Bedingungsvarianten und ihr Zusammenhang mit allen beobachteten Reihenfolgen analysiert werden. Abgesehen davon daß das Korpus gar nicht Belege für alle Bedingungskonstellationen enthält, wäre das aus Zeitgründen von vornherein nicht zu bewältigen. Es würde nämlich, setzt man für die Bearbeitung einer Bedingunsgkonstellation eine Sekunde an, bei angenommener gegenseitiger Unabhängigkeit aller Bedingungen rund zweihundert Jahre dauern, bis alle einmal betrachtet worden sind. Wie artifiziell diese Kalkulation ist, wird schon klar, wenn man einräumt, daß die Betrachtung einer Bedingungskonstellation leicht auch durchschnittlich fünf Sekunden kosten kann, womit die Gesamtdauer auf ein Jahrtausend anwachsen würde. Immerhin: diese Betrachtungen lassen ahnen, in welchen Dimensionen man sich bewegt, zumal, wenn man bedenkt, daß die Analyse mit der puren Betrachtung einer Bedingungskonstellation ja keineswegs getan ist, wie gleich noch ausgeführt wird. Einstweilen kam es nur darauf an, die Komplexität des im Spiel befindlichen Bedingungsrepertoires

erkennbar zu machen. Bleibt nur noch anzumerken, daß in einem Analyseschritt klarerweise nie alle Bedingungen einbezogen sein können; vielleicht zwei bei Konstant- oder Gleichhaltung der anderen, im nächsten Schritt dann vielleicht zwei oder drei andere oder nur eine; wieviele und welche das jeweils sein müssen, ergibt sich aus dem vorherigen Analyseschritt, und das ist der Punkt: Der Verlauf der Exploration ist - anders als bei einer experimentellen Hypothesenüberprüfung - vorher nicht genauer zu bestimmen als eben durch die Festlegung auf die als relevant angenommenen Bedingungen und die wenigen bis dato bekannten Abhängigkeiten zwischen ihnen. So ist es ökonomisch und für den kontinuierlichen Gang der Analyse wesentlich, spontan Zugriff auf möglichst alle Bedingungsvarianten zu haben, die sich im Korpus finden. Und das zu organisieren, erfordert eine Datenbank. Analysen wie die hier skizzierte wären ohne Einsatz eines solchen Instruments nicht durchzuführen[5)].

2.2. Andere Gründe

Während die Komplexität, ist sie einmal in dem beschriebenen Maß anvisiert, ein Umstand ist, der zum Einsatz einer Datenbank zwingt, sind andere Umstände eher begleitender Art. Sie könnten auch ohne Datenverarbeitungsinstrumente bewältigt werden. Gemeint sind die Zeitintensität solcher Analysen, die Fehleranfälligkeit und das Erfordernis flexibler Erweiterung und Umorganisation. Zeitintensiv sind Analysen von größeren Korpora einfach schon deshalb, weil laufend andere Belege in die Betrachtung einbezogen werden müssen, um aktuell aufscheinende Zusammenhänge zu überprüfen. Ist man bei diesem Geschäft auf Karteikarten oder Transkriptionen verwiesen, kostet die Suche pro Beleg exponentiell mehr Zeit als in einer Datenbank.

Zum zweiten sind manuelle Suchen eher fehleranfällig; es werden Belege übersehen, und wenn es gerade wichtige waren, geht die Analyse unter Umständen eine Zeitlang in die falsche Richtung und muß später, wenn es gutgeht und die falsifizierenden Belege noch auftauchen, revidiert werden. Schließlich sind Analysen wie die hier durchzuführenden in einem bestimmten Sinn offen. Es kann sich im Zuge eines Analyseschritts ergeben, daß Kategorien zusammengefaßt werden können, entfallen können oder neue Eigenschaften dieser oder jener Segmentklassen nachträglich einbezogen werden müssen. Solche Veränderungen sind umso leichter und sicherer zu organisieren, je flexibler das jeweilige Dokumentationsinstrument ist; und Transkriptionen, Karteien oder Merkzettel sind infolge ihrer physikalischen Gegebenheiten schlecht umzuorganisieren, von der auch hierbei wiederum gegebenen Fehleranfälligkeit ganz zu schweigen.

---

5) Man mag sich fragen, wie solche Analysen ohne Datenbanken bewältigt wurden. Sie wurden es eben nicht, zumindest nicht für hinreichend große Korpora von Äußerungen; so erklärt sich auch Behaghels bis heute zutreffende Klage, daß das Zusammenwirken der Bedingungen zu wenig erforscht ist. Wurden die Analysen aber trotzdem komplexer angelegt, so mußten aus Kapazitätsgründen eben die Belegzahlen niedrig oder die Analysen auf wenige Konstellationen beschränkt bleiben, wie in Boost (1957) oder Reichel (1987), um mindestens eine sprachwissenschaftliche und eine psychologische Studie zu nennen.

Nicht so sehr mit der Sache sondern mit der Dynamik von Arbeitsgruppen hat die letzte Überlegung zu tun: Es ist eine Sache der Erfahrung, daß die Ordnung von Karteikästen oder vergleichbaren Mitteln eben infolge ihrer mangelnden Flexibilität sich über kurz oder lang auflöst und nicht mehr von allen durchschaut wird, die damit arbeiten müssen. Es steigen die Zahlen von individuellen Notizen und Zettelchen, Zeichen und Markierungen und zwar unsystematisch. Das ist im Fall der Anwendung einer Datenbank allein durch den höheren Grad an Vernetztheit ihrer Tabellen, Felder und Werte drastisch eingeschränkt. Eine Datenbank wirkt also auf längere Zeit für alle ihre Nutzer als ein stärkeres äußeres Bindeglied.

3. Die Struktur der Daten

Ob man Wilhelm Dilthey's Lehre vom wesentlichen Unterschied zwischen den Geistes- und den Naturwissenschaften zu den gelungensten Beiträgen zur Wissenschaftsgeschichte rechnet oder nicht - sie prägt bis heute die Sicht, genauer die Grenzen der Sicht, auf beiden Seiten, und da der Computer als ein typisch naturwissenschaftliches Produkt und Instrument gesehen wird, werden die Chancen seiner erfolgreichen Nutzung im geisteswissenschaftlichen Feld von vornherein auf beiden Seiten mehrheitlich als gering veranschlagt. Nun können wir zwar im Hinblick auf die allgemeine wissenschaftliche Zielsetzung keinen wesentlichen Unterschied zwischen Geistes- und Naturwissenschaften erkennen: Jede wissenschaftliche Forschung strebt danach, die Welt innerhalb und außerhalb des Menschen besser zu verstehen und explizit zu beschreiben, was letztlich bedeutet, die Vielfalt der Erscheinungen auf allgemeine Zusammenhänge zu beziehen und das menschliche Wissen darüber zu vermehren[6]. Dennoch kann nicht bestritten werden, daß naturwissenschaftliche und technische Forschung aufs Ganze gesehen auf den Einsatz von Rechnern besser vorbereitet ist als die geisteswissenschaftliche. Stark vereinfachend, im Kern aber sicher zutreffend, kann man das damit erklären, daß unsere Kenntnis von der Natur in genaueren, einheitlicheren und formaleren Theorien vorliegt, unsere Kenntnisse von Vorgängen und Ergebnissen der menschlichen Geistestätigkeit hingegen noch weniger genau, weniger einheitlich und weniger formalisierbar sind - was natürlich mit der relativen Jugend der Geisteswissenschaften zu tun hat[7].

Im Einzelfall tritt diese Kluft zwischen den Eigenschaften und Bedingungen des Instruments einerseits und den Voraussetzungen in den Geisteswissenschaften andererseits zutage, indem es dem geisteswissenschaftlichen Anwender schwerfällt, seine Bearbeitungserfordernisse so zu beschreiben, daß sie sich in Begriffen von Bedingungen und Möglichkeiten von Hard- und besonders Software reformulieren lassen. Im vorliegenden Fall ist das geradezu augenfällig. Sind die Fragen, ob und welche Datenbanksoftware angewendet werden soll, einmal grundätzlich positiv entschieden, stellen sich vor der Realisierung unausweichlich eine Reihe von Fragen, die Gegebenheiten des Datenbankmodells, des ausgewählten

6) Einläßlicher gehen auf diesen Punkt Dietrich/zu Putlitz (1988) ein.

7) Einen wissenschaftsgeschichtlichen Nachweis darüber erbringt z.B. Joachim Ritter (1974).

Programmpakets und örtliche Gegebenheiten zum Ausgangspunkt haben. Eine zentrale Frage ist die nach den Datenstrukturen. Wie in Abschnitt 4 dargestellt wird, hat ORACLE - wie andere Datenbankprogrammsysteme auch - eher Eigenschaften eines Werkzeugkastens und nicht eines fertigen Aktenschranks; der wird eben mit den Instrumenten des Systems aufgebaut. Dabei hat man mancherlei Ziele im Auge, solche den Komfort, die Sicherheit und die Leistung betreffend, allen voran aber eines, das Verhältnis zwischen der Struktur des im Projekt bearbeiteten Beobachtungsfeldes (technisch: der Daten) und der Struktur der aufzubauenden Datenbank zu definieren. Im Grunde muß letztere sich aus der ersteren ergeben, und damit sind wir beim Problem. Welche Strukturen in den Projektdaten sind und wie sie zu fassen sind, liegt nämlich keineswegs immer auf der Hand. In Projekten, in denen die Ergebnisse auf dem Weg der schrittweisen Datenanalyse gesucht werden, wie im hier gegebenen Beispiel, kommt noch die im letzten Abschnitt schon angesprochene Schwierigkeit hinzu, daß sich die Antwort auf die Frage, welche Merkmale und Eigenschaften des zu analysierenden Beobachtungsfeldes für die weitere Bearbeitung und damit für den Entwurf der Datenbank bedeutsam sind, erst durch die Analyse selbst ergibt und nicht vorher schon festliegt. Diese müssen aber soweit wie möglich angegeben werden können, wenn die Datenbank nicht nur triviale Eigenschaften der Daten widerspiegeln soll. So bleiben denn als Anhaltspunkte für die Ermittlung der Strukturen in den Daten zunächst einmal (a) die Begriffe, die in der Projektfragestellung enthalten sind, (b) Kategorien, die infolge von Aussagen in der einschlägigen Forschung als bedeutsam anzusehen sind und (c) alle Eigenschaften der sprachlichen Belege und der sonstigen empirischen Gegebenheiten, die in die Analyse eingehen und nicht von vornherein aus irgendwelchen evidenten Gründen als irrelevant gelten müssen. Uns ist kein allgemein verbindliches und sicheres Verfahren bekannt, nach dem vorzugehen wäre, um für ein beliebiges sprachwissenschaftliches Projekt dieser Art die Struktureigenschaften anzugeben, die für den Aufbau einer Datenbank jeweils maßgeblich sind. Wir halten uns also einfach an die oben entwickelten drei Gruppen von Anhaltspunkten. Hat man erst einmal diese oder jene Struktureigenschaft als relevant erkannt, verbinden sich damit vor dem Hintergrund des im jeweiligen Wissenschaftsfeld vorhandenen Repertoires an Kategorien rasch Vermutungen in Richtung auf mögliche weitere Kandidaten - nach unserer Erfahrung mehr als man sich dann schon zu beurteilen zutraut.

Die Fragestellung des vorliegenden Projekts läßt sich, wie im ersten Abschnitt schon kurz referiert, (a) auf die knappe Formel bringen: Welche Zusammenhänge bestehen zwischen pragmatischen Umständen der Äußerungssituation sowie semantischen Eigenschaften der Äußerung einerseits und der Wortstellung in der Äußerung anderseits? Aus der einschlägigen Forschung wissen wir (b), daß der Spielraum für die Wortstellungen bei einer gegebenen Struktur durch syntaktische Regeln begrenzt ist, indem syntaktische Eigenschaften der Äußerungsteile eben das Ausmaß ihrer Beweglichkeit in der Äußerung tangieren und daß das alles für verschiedene Sprachen verschieden ist. Des weiteren ist aus dem Forschungsstand natürlich abzuleiten, welche pragmatischen Umstände, welche semantischen Merkmale und welche Arten von Äußerungsteilen als stellungssensitiv bzw. stellungsvariabel in Betracht kommen. Davon geleitet haben wir dann Personen veranlaßt, (c) in von uns vorstrukturierten Situationen und zu ausgewählten Anlässen zu sprechen, d.h. Äußerungen spontaner, mündlicher Rede zu produzieren, die auf Tonband aufgenommen wurden.

Durch diese Darstellung scheint die eine oder andere Struktur unserer Daten schon hindurch. Ein Verfahren, dies deutlicher herauszuarbeiten, setzt gewissermaßen am Ende an. Es ergibt sich, indem man sich vor Augen führt, zu welchen Ergebnissen die Projektarbeit gelangen soll. Das sind im vorliegenden Fall eben Aussagen der Form:

> Wenn ein Ausdrucksteil mit der syntaktischen Eigenschaft s unter dem pragmatischen Umstand p und mit der semantischen Eigenschaft i in der Äußerung u des Textes t auftritt, dann nimmt er die absolute/relative Position n ein.

In der Realität werden die Regularitäten so einfach nicht zu beschreiben sein; wir müssen sicher mit komplexeren Bedingungsbeschreibungen rechnen. Das Schema der Einzelregeln darin wird aber doch im wesentlichen so gebaut sein, und wenn man zudem ausnutzt, daß die darin enthaltenen Ausdrücke

(a) "syntaktische Eigenschaft s",
(b) "pragmatischer Umstand p",
(c) "semantische Eigenschaft i" usw.

durch

(a) "Element der Klasse s der syntaktischen Kategorie S",
(b) "Umstand der Art p bezogen auf die pragmatische Dimension P,
(c) "Merkmal i bezogen auf die semantische Kategorie I" usw.

paraphrasiert werden können, dann erkennt man, daß sich das vereinfachte Regelschema als Hilfsmittel einsetzen läßt bei dem Versuch zu entscheiden, ob ein als relevant aufscheinender Begriff für die Beschreibung der Struktur im Beobachtungsfeld bedeutsam ist, damit in die Bestimmung der Datenstruktur eingehen muß und ob es sich gegebenenfalls um eine Kategorie oder um eine Ausprägung (einen Wert einer Kategorie) handelt. Wo bei Anwendung dieser Hilfsprozedur Zweifel bleiben, lassen sie sich jedenfalls nur durch weitere begriffliche und/oder theoretische Sondierungen seitens des Anwenders beseitigen, keinesfalls durch Rücksicht auf Gegebenheiten der Datenbanksoftware[8]. Damit ist das Ziel in Sicht. Es muß jetzt nämlich für jeden Bestandteil des genannten Regelschemas bestimmt werden, welche Kategorien, Dimensionen, Klassen, Arten und Merkmale in den Daten vorhanden sind und bedeutsam werden können und welche Zuordnungsbeziehungen zwischen ihnen bestehen, ob beispielsweise zwei gleichlautende Äußerungen zu zwei verschiedenen Texten gehören oder nicht. Wie mehrfach dargelegt, werden die Ergebnisse also letztlich Aussagen über Teile sprachlicher Äußerungen, genauer über ihr Stellungsverhalten in der linearen Kette der Äußerung sein, in der sie stehen. Nun sind in einer natürlichen Sprache bekanntlich nicht alle Segmente einer linearen Kette gleichermaßen frei verschiebbar. Die Buchstaben eines Wortes sind es zum Beispiel überhaupt nicht, ganze Wörter schon eher, Wortgruppen oder Satzglieder in einer Äußerung unterschiedlich, ebenso Sätze innerhalb des Textes. Die einschlägige Forschung hat sich bislang in erster Linie mit der Stellung von Satzgliedern innerhalb der Äußerung befaßt. Darüber ist am meisten bekannt, und

---

8) Es wird sicherlich Leser dieses Hefts geben, denen all das bis zur Stufe der Selbstverständlichkeit geläufig ist; sie bitten wir um Nachsicht.

da setzt auch das vorliegende Projekt an. Mithin sind die kleinsten Segmente, mit denen wir es zu tun haben, Satzglieder, im folgenden im Blick auf ihre syntaktische Seite als "Phrasen" bezeichnet.

Phrasen, ihre Stellung und ihre Eigenschaften bilden also offenbar eine Ebene der Strukturierung in dem gesamten Analysefeld. Die absolute Stellung einer Phrase in einer Äußerung wird durch die Angabe der Positionen ihrer Wortformen[9] erfaßt, wobei die Wortformen von Anfang bis Ende der Äußerung mit kontinuierlich aufsteigenden ganzen Zahlen numeriert werden. Des weiteren gilt beispielsweise als sicher, daß die Stellung einer Phrase relativ zu anderen Phrasen geregelt ist und zwar (a) relativ zum finiten (d.h. konjugierten) Verbteil und (b) relativ zu anderen Phrasen der Äußerung, zu denen syntaktische Abhängigkeiten bestehen. Auch dies muß also als strukturierendes Merkmal der Daten gelten. Des weiteren ist allgemein bekannt, daß die syntaktische Funktion einer Phrase (Subjekt, Objekt, adverbiale Bestimmung, Apposition u.a.) sowie ihre syntaktische Kategorienzugehörigkeit (Substantivgruppe, Verbalgruppe u.a.) stellungsrelevant sind. Ferner gibt es klare Hinweise darauf, daß auch Bedeutungseigenschaften einer Phrase Einfluß auf ihre Stellung nehmen können, und zwar gleich mehrere Arten von Bedeutungseigenschaften, zum Beispiel die, ob sie einen Ort, eine Zeit, einen Zustand oder Vorgang, ein Ding bezeichnet, oder die, ob das zentrale Wort der Ding-Phrase etwas Belebtes bedeutet oder nicht, und schließlich auch die, in welchem inhaltlichen Verhältnis die in einer Subjekt- Objekt- oder Adverbphrase bezeichneten Dinge zum Vorgang oder Ereignis stehen, das im Verb bezeichnet wird, ob die Phrase also einen Urheber oder einen Betroffenen des Vorgangs oder Ereignisses bezeichnet oder ein Instrument oder einen Empfänger, einen Ausgangspunkt, ein Ziel oder anderes. Schließlich ist gezeigt worden, daß es auf die Stellung wirkt, ob die mit einer Phrase bezeichnete Sache zum erstenmal im Text bezeichnet oder beispielsweise wiedererwähnt wird.

Man erkennt nun sicher leicht, daß die Phrasen Segmente in den Daten bilden, die mit ihren diversen Eigenschaften deutlich einen Strukturbereich konstituieren, dessen Konturen hier nur kursorisch vorgeführt wurden. Im einzelnen sind die auf dieser Ebene als relevant erachteten stellungssensitiven und stellungsvariablen Kategorien im fünften Abschnitt und seinen Listen genannt. Insgesamt mag danach der Eindruck entstehen, als wäre eigentlich doch schon ausgemacht, wovon nun die Stellung einer Phrase in einer Äußerungskette abhängen kann. Dieser Eindruck wäre unzutreffend. Zum einen ist das Repertoire der beteiligten Kategorien sicherlich nicht vollständig, und ebensowenig sind es die einstweilen unterschiedenen Ausprägungen je Kategorie. Die Analyse kann und wird hier zu Modifikationen führen und damit steht die gesamte Theorie zur Disposition, wie übrigens in der Forschung immer. Und genau zu diesem Zweck, zum Zweck der Analyse auf der unmittelbaren Beobachtungsebene vor aller Kategorisierung, muß auch der

---

9) Eine Phrase besteht aus einer oder mehreren Wortformen, die ihrerseits zunächst rein äußerlich als die zusammenhängenden Buchstabenketten zwischen Leerzeichen (oder Leerzeichen und Satzzeichen oder....) verstanden werden können. Hier wird - nebenbei bemerkt - deutlich, daß beim Transkribieren der Tonbandaufzeichnungen schon strukturierende Entscheidungen fallen, denn die Leerzeichen werden vom Transkribierenden eingesetzt und die Satzzeichen auch.

Wortlaut jeder einzelnen Phrase und jeder Äußerung und des ganzen Textes[10] in der Datenbank vorhanden sein. Soviel zum Strukturbereich der Phrasen.

Eine weitere Ordnungsebene läßt sich aus dem Umstand ableiten, daß Phrasen sich zu größeren sprachlichen Einheiten, eben den Äußerungen verbinden. Äußerungen sind aus mehreren Gründen besonders bedeutsam. Sie sind es nämlich, die den Operationsbereich der Stellungsregeln bilden. Alle hier interessierenden Stellungsregularitäten betreffen also die Stellung von Phrasen in Äußerungen. Zum zweiten sind Äußerungen als Einheiten relevant, weil einige Äußerungseigenschaften direkt in Stellungsregularitäten eingehen, z.B. die, ob mit der Äußerung eine Aussage, eine Frage, eine Aufforderung oder ein Wunsch ausgedrückt wird, ob sie ihrerseits innerhalb größerer thematischer Einheiten wie Abschnitte oder Paragraphen am Anfang oder nicht am Anfang steht. Was auf Äußerungsebene im einzelnen zu erfassen ist, wird ebenfalls in Abschnitt 5 im Detail aufgeführt.

Schließlich bilden Äußerungen einen Text, ebenfalls eine sprachlich relevante Einheit mit Eigenschaften, die sich auf die Stellung von Phrasen in Äußerungen auswirken. Zu diesen Eigenschaften gehört in erster Linie natürlich, um welche Sprache es sich handelt, des weiteren, welcher Art die globale Fragestellung - die Diskursaufgabe, wie man auch sagt - ist, der der Text gilt[11], von welcher Sache in einem Text die Rede ist (z.B. von der Einrichtung eines Zimmers, von einem Weg, von einem Reifenwechsel oder einem Autounfall), ob Sprecher und Hörer sich gerade sehen können oder nicht (Telephonbedingung), ob beiden oder einem oder keinem der Sachverhalt vor Augen ist, über den gesprochen wird und anderes mehr. Es ist klar, daß Texte Einheiten sind, die mit ihren diversen stellungssensitiven Eigenschaften zur Struktur des Beobachtungsbereichs beitragen. Wir haben diese Ebene des Textes und seiner Eigenschaften (inklusive situativer Produktionsbedingungen) die Designebene genannt. So ergeben sich also die Phrasen, die Äußerungs- und die Designebene mit ihren jeweiligen stellungssensitiven und stellungsvariablen Kategorien bzw. Einheiten als die Bereiche und Elemente, die die Struktur des Beobachtungsbereichs ausmachen. Damit wäre alles erwähnt, wenn nicht noch eine Mißlichkeit zu berücksichtigen wäre.

Könnte man sicher sein, daß alle Stellungsgesetzmäßigkeiten einer Sprache in jedem beliebigen Text eines Designs belegt sind, dann würde es genügen, zu jedem interessierenden Design genau einen Text zu haben und zu analysieren, denn er müßte ja alle gesuchten Regularitäten mindestens einmal exemplifizieren, so wie ein Goldatom die Struktureigenschaften aller Goldatome instanziiert. Um diese zu analysieren, würde man ja auch nicht zwanzig Zentner Gold durch das Rasterelektronenmikroskop schieben. So ist es aber in der Sprache nicht. Man kann nicht erwarten, daß sich in einem Text pro Design alle Stellungsregeln

---

10) Der Wortlaut des Textes läßt sich allerdings in unserem Fall aus dem Wortlaut der Äußerungen und einer Reihenfolgeinformation einfach und vollständig automatisch rekonstruieren, nicht so die Äußerung aus den Phrasenwortlauten, weil nicht alle Bestandteile von Äußerungen als Phrasen in unserem Sinne gelten und demnach nicht erhalten werden.

11) In Beschreibungen gelten andere Regeln für die Verteilung der Information in der Äußerung als in Erzählungen und Instruktionen; vgl. dazu die Projektanträge der Teilprojekte A2 und A1 des SFB 245.

offenbaren. Also muß man mehrere Texte pro Design untersuchen. Die gewinnt man, indem man statt einer eben mehrere Personen bittet, im Rahmen desselben Designs zu sprechen, d.h. je einen Text zu produzieren. Damit ist aber nicht nur ein Problem gelöst, sondern auch ein weiteres geschaffen. Nach allem, was wir über die Struktur der Sprache von Sprachgemeinschaften wissen, kann man nämlich nicht sicher sein, daß zwei beliebige Personen dieselben Stellungsregeln anwenden. Das wird einem sofort einsichtig, wenn man einen Kleisttext über das Erdbeben von Chile mit einem Zeitungsbericht darüber vergleicht. Man muß mit sprecherabhängiger Sprachvariation rechnen und für die Analyse demnach die Möglichkeit schaffen, sprecherspezifische Regelhaftigkeiten durch den Vergleich der Belege eines Sprechers mit den Belegen anderer oder anderer Sprechergruppen überprüfen zu können. Auch dies bringt also Struktur in das Beobachtungsfeld. Im Unterschied zu den bisher beschriebenen Strukturkategorien ist diese hier aber keine, die man begrüßt und die man beabsichtigt hat, sondern eine, die man, sobald sie erkannt ist, wieder "rausrechnen" und in anderen Fragestellungen weiterverfolgen muß - wie immer man das bewerkstelligen kann. Aber gerade dazu muß man sie zunächst einmal identifiziert haben; also muß der Sprecher bzw. der individuelle Text pro Design erfaßt und diese Information bis auf die Phrasenebene transportiert werden. Mehr auch darüber in den folgenden Abschnitten zur Realisierung der Struktur in der Datenbank und zu den Arten ihrer Nutzung.

4. Begründung für ORACLE

Hier soll dargelegt werden, warum wir uns für die Benutzung des Datenbanksystems ORACLE der Firma ORACLE Deutschland GmbH entschieden haben. Zunächst werden im Abschnitt 4.1 drei verschiedene Datenstrukturmodelle vorgestellt, und es wird gezeigt, warum wir das relationale Datenmodell präferieren. Welche Gesichtspunkte uns dazu veranlaßt haben, die Software ORACLE als Realisierung einer relationalen Datenbank auszuwählen, soll in Abschnitt 4.2 gezeigt werden.

4.1. Darstellung möglicher Datenbank-Strukturen

Würde die Aufgabe zur Verwaltung von Daten darin bestehen, vorhandene Daten nur zu erfassen und abzuspeichern, so würde es genügen, einen komfortablen Editor zu verwenden. Solange die Menge der Daten bestimmte Grenzen nicht überschreitet und keine komplexen Fragestellungen auftauchen, liefern Suchbefehle auf Textteile schnell und einfach das gewünschte Ergebnis. Beim Arbeiten mit den Daten des Projekts "Äußerungsaufbau" geht es aber vor allem darum, die Beziehung zwischen verschiedenen Daten zu erfassen und Suchen mit komplexeren Verknüpfungen durchzuführen. Eine solche Suche könnte z.B. für unsere Daten lauten: Suche alle Wortlaute von Phrasen mit den Bedingungen 'Diskurstyp ist Beschreibung', 'der Informant spricht Deutsch und beschreibt das Objekt Puppenstube' und 'die syntaktische Funktion der Phrase ist nicht Subjekt'. Solche Suchen sind in Editoren meist nicht realisiert. Sie zu programmieren ist aufwendig und hat meist zur Folge, daß die Daten z.B. spaltengerecht oder in verschlüsselter Form - wie

z.B. bei AQUAD[12] - eingegeben werden müssen. Textverarbeitungssysteme sind aus denselben Gründen wie Editoren für die Darstellung von Beziehungen zwischen Daten nicht geeignet. Datenbanken bieten das gewünschte, nämlich die Datenbeziehungen aus der Sicht eines Anwenders abzubilden.

Da die physikalische Datenorganisation[13] aus praktischen Gründen der Realisierung im Computer oft sehr weit entfernt ist von der logischen Datenstruktur aus der Sicht des Anwenders, ist es notwendig, beides zu trennen, d.h. die Benutzersicht sollte - so weit möglich - ohne Rücksicht auf die physikalische Darstellung formulierbar sein. Die Software soll also im Idealfall die Übersetzung der logischen Struktur der Daten in eine effiziente physikalische Struktur gewährleisten.

Es gibt im wesentlichen 3 verschiedene Datenmodelle, die von der heutigen Datenbank-Software als Grundlage verwendet werden (vgl. u.a. Martin, 1987, Kap. 8 und 9):

a) hierarchisches Datenmodell oder Baumstruktur,
b) Netzwerkdatenmodell oder Plexstruktur,
c) relationales Datenmodell.

a) Baumstruktur

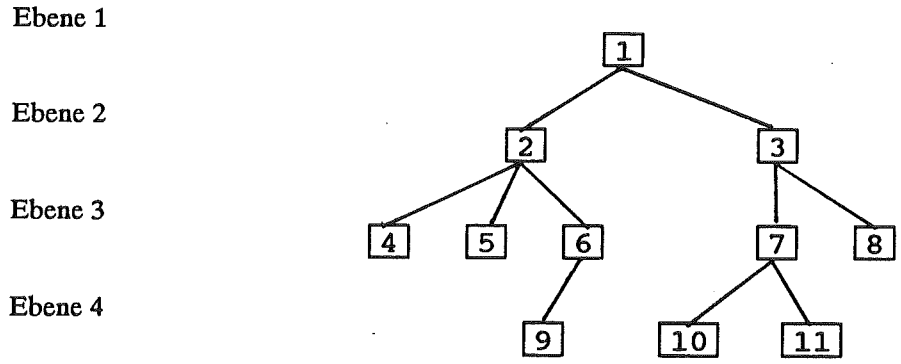

Abbildung 1: *Hierarchische Struktur*

Hierarchische Datenmodelle setzen eine vereinfachte Sicht der durch die Datenbank repräsentierten Realität voraus. Sie fordern, daß die Elemente der Datenbank nur über streng hierarchische Beziehungen verknüpft sind. Jedes Element - außer dem höchsten - hat genau ein übergeordnetes Element, aber möglicherweise mehrere untergeordnete Elemente. Eine solche hierarchische Beziehung wird mathematisch als Beziehung vom Typ 1:n bezeichnet, wobei n eine natürliche Zahl ist. (Obiges Beispiel einer Baumstruktur hat

---

12) AQUAD ist ein Programmpaket in Turbo Prolog für die computerunterstützte Analyse qualitativer Daten.

13) Die physikalische Datenorganisation ist vereinfacht gesagt die Anordnung der Bytes auf einem Datenträger und die daraus resultierenden Zugriffsmechanismen. Dies ist in erheblichem Maß von der Art der eingesetzten Hardware abhängig (vgl. u.a. Gillner, 1984).

z.B. in der höchsten Ebene eine Beziehung von 1:10, d.h., das Element (1) hat 10 untergeordnete Elemente, nämlich die Elemente (2)...(11).) Die Nachteile dieser Struktur liegen zum einen in der Schwierigkeit, die Realität abzubilden und damit Redundanz (mehrfaches Speichern derselben Daten) im Datenbestand in Kauf nehmen zu müssen, zum anderen in der Länge der Zugriffswege bei tiefen Bäumen.

b) Netzstruktur

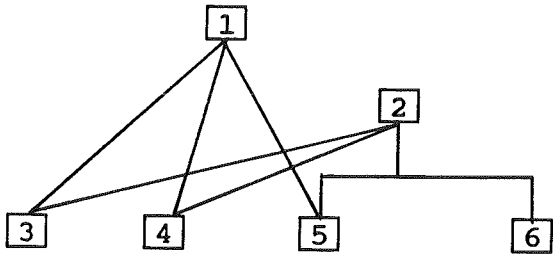

Abbildung 2: *Netzstruktur*

Im Netzwerkdatenmodell hat jedes Element - außer dem höchsten - mindestens ein übergeordnetes Element und möglicherweise mehrere untergeordnete. Diese Struktur erlaubt Beziehungen zwischen den einzelnen Elementen vom Typ m:n. Ein derartiges Datenmodell ist ausreichend flexibel, um alle in der Realität vorkommenden Abhängigkeiten zwischen den Objekten in der Datenbank abbilden zu können. Trotzdem bleibt der Nachteil von langen Zugriffswegen bei komplexen Netzen.

Bei den beiden Datenmodellen Baumstruktur und Netzstruktur ist in der Realisierung der Zugriff auf ein Element prinzipiell nur über das übergeordnete Element möglich. Sollen die Vorteile dieser Datenmodelle genutzt werden, setzt das die Kenntnis der physikalischen Struktur voraus. Eine Verlängerung der Suchzeiten bei langen Kettenstrukturen ist fast unumgänglich. Es gibt stattdessen eine einfachere und elegantere Methode:

c) relationales Modell

Der relationale Ansatz basiert darauf, die Ergebnisse der mathematischen Theorie der Relationen auf die Datenbankproblematik zu übertragen. Die Elemente eines Sachverhalts und ihre Beziehungen untereinander werden als Relationen betrachtet. Hat man diesen Schritt einmal vollzogen, so kann man das ganze Instrumentarium aus der Theorie der Relationen übernehmen und damit arbeiten. Daraus ergibt sich die Darstellung der Relationen in Tabellenform sowie der Operationen auf diesen Relationen als Operationen auf den Tabellen. Eine Tabelle besteht aus n-Tupeln von Elementen (Zeilen der Tabelle) mit jeweils m Spalten.

	RELATION	SPALTE1	SPALTE2	SPALTE3	.....	SPALTEm
Tupel(1)						
Tupel(2)						
Tupel(3)						
Tupel(4)						
:						
Tupel(n)						

Abbildung 3: *relationale Struktur*

Das einzige benötigte Basiskonzept ist das der Relation. Operationen auf Relationen erzeugen wieder Relationen. Auf die Relationen kann man die Regeln der Normalisierung (Begriff für eine formale Vorgehensweise, auf die in Abschnitt 5 näher eingegangen wird) anwenden, welche wesentlich sind, um einen sinnvollen Aufbau einer Datenbank ohne Redundanz und mit hoher Datenintegrität (der Gewähr, korrekte und vollständige Daten zu haben) zu erreichen. Die beschriebenen Prinzipien für das relationale Modell beziehen sich ausschließlich auf die logische Beschreibung der Datenstruktur. Die relationale Sicht der Daten muß nicht auf die physikalische Organisation angewendet werden, d.h. es gibt viele Möglichkeiten, in einem Computer die physikalische Darstellung einer solchen relationalen Datenbank zu implementieren.

Ziel des Einsatzes des Hilfsmittels Datenbank im Projekt ist es, ausgehend von den bis dato bekannten Abhängigkeiten der Wortstellung in einem Satz vorher nicht genauer festlegbare weitere Abhängigkeiten und Beziehungen aufzudecken. Dies bedeutet im Relationenmodell die Einführung beliebiger neuer Relationen zwischen den gegebenen Kategorien. Ein relationales Datenbanksystem ist offen für das Zusammenfassen, Entfernen alter und Einführen neuer Kategorien, wie es in Abschnitt 2 für den Verlauf der Exploration gefordert wird.

4.2. Warum wurde ORACLE als Realisierung einer relationalen Datenbank gewählt?

Für ein wirklich relationales Datenbank-System gilt, daß es jederzeit in der Lage sein muß, Datenbanken vollständig mit Hilfe seiner relationalen Fähigkeiten verwalten zu können.[14] Entscheidungskriterien für ein bestimmtes relationales Datenbanksystem waren folgende Gesichtspunkte (vgl auch Diemer, 1989, S. 117f.):

* Dasselbe System soll unter mehreren Betriebssystemen lauffähig sein und zwar, soweit wie möglich, identisch und mit der Möglichkeit des unproblematischen Austauschs von Daten. Ein Konzept nur für den Großrechner ist nicht akzeptabel, da die Verfügbarkeit bei den PCs größer und die Performance gerade bei Ein-/Ausgaberoutinen besser ist. Eine reine PC-Lösung scheidet aufgrund der Menge der Daten aus. Damit ist gemeint, daß sowohl der Platzbedarf die Grenzen eines normalen PCs sprengen würde, als auch die zu erwartende Reduzierung der Performance bei Suchen diese Lösung ad absurdum führt. Ferner ist ein gewünschtes Endziel, eine einzige vollständige Datenbasis zu haben, auf die alle Projektmitglieder Zu-

14) Dies gilt z.B. nicht für dBASE IV, auch wenn es relationale Ansätze hat.

griff haben, wenn möglich auch gleichzeitig. Dies gestaltet sich bei PCs, die untereinander nicht vernetzt sind, als schwierig.

* Es muß die Möglichkeit geben, alle in der Datenbank gespeicherten Informationen abzurufen.

* Ein Benutzer der Datenbank soll sich ausschließlich mit der Datengewinnung und -manipulation beschäftigen können. Es muß also eine Sprache geben, mit der man interaktiv Fehler in den Datenbankanweisungen beseitigen und die aus der Datenbank ableitbaren Relationen abfragen kann. Diese Sprache sollte deskriptiv sein und nicht prozedural wie z.B. viele Programmiersprachen.

* Einfügung, Aktualisierung und Löschen soll für mehrere Datensätze gleichzeitig möglich sein.

* Es muß möglich sein, nicht vorhandene Information unabhängig vom Datentyp, also z.B. numerische oder alphanumerische Daten, als Nullwert darzustellen.

* Eine Schnittstelle zu einem Textverarbeitungssystem ist wünschenswert.

* Statistikprozeduren sollten im System integriert sein.

* Für eine spätere Ausbaustufe wäre das Konzept der Verteilungsunabhängigkeit von Vorteil. D.h. in der einfachen Form, daß eine Datenbank auf dem Großrechner lokalisiert ist, daß aber von jedem PC, der mit dem Großrechner (ORACLE-gemäß) vernetzt ist, auf die Datenbank zugegriffen werden kann, als wäre sie auf dem PC selbst. Dieses Konzept wird auch als Client/Server-Architektur bezeichnet. In einer komplexeren Form von Verteilungsunabhängigkeit ist es sogar möglich, daß Teile einer einzelnen Datenbank auf unterschiedlichen Rechnern lokalisiert sind.

Das Datenbanksystem ORACLE gliedert sich in mehrere Komponenten, wobei die wichtigsten der Datenbankkern, das RDBMS (Relational Database Management System), und SQL*Plus, ein Dialogprogramm als Schnittstelle zwischen Benutzer und dem RDBMS, sind. ORACLE bietet den (fast) identischen Ablauf auf PCs und Großrechnern sowie Workstations. Eine Schnittstelle zu WordPerfect, Routinen zur Berechnung von Mittelwert und Varianz, sowie Tabellenkalkulation sind vorhanden. Das Erlernen einer prozeduralen Sprache zur Datengewinnung und Datenmanipulation ist nicht nötig. (Hierzu bietet ORACLE die Sprache SQL - Structured Query Language an.) Abgesehen davon, daß kein anderes damals zur Diskussion stehendes Datenbanksystem den identischen Ablauf auf dem IBM-Großrechner und MS-DOS PCs zu diesem Zeitpunkt anbot, sprachen für ORACLE die lokalen Gegebenheiten:

(a) Einsatz auf dem IBM-Großrechner 4381-12 des Universitätsrechenzentrums Heidelberg unter dem Betriebssystem VM/CMS (Rel. 5) mit ORACLE Vers. 5.1, SQL*Plus Vers. 2.1 und SQL*Forms Vers. 2.2.

(b) Campuslizenz für PCs unter dem Betriebsystem MS-DOS mit ORACLE Vers. 5.1B, SQL*Plus Vers. 2.1 und SQL*Forms Vers. 2.3.

Es gab ferner ein weiteres Projekt in Heidelberg, PROTEXT, welches sich mit einer erweiterten Abfragesprache für ORACLE beschäftigte, deren Nutzung man in Erwägung zog. ORACLE ließ in der damaligen Version das Client/Server-Modell zu, also die Lokalisation einer Datenbank auf einem anderen Rechner, aber nicht das Konzept der Verteilung einer einzelnen Datenbank. Dieser Nachteil wurde aber bei der Betrachtung nicht als schwerwiegend angesehen. Mittlerweile ist dieses Konzept für die neueren Versionen von ORACLE angekündigt.

Als Alternative zu ORACLE wurde damals folgende Kombination diskutiert: dBase, ein weit verbreitetes Datenbanksystem (nur) für PCs, und DB2, das IBM-Datenbanksystem auf dem Großrechner IBM 3090-180 des Rechenzentrums unter dem Betriebssystem MVS-XA. Eine hundertprozentige Entsprechung der beiden Datenbanksysteme ist aber nicht gegeben. Die Verwaltung und Verwendung von sehr langen Textketten (mehr als 240 Zeichen) ist in DB2 besser gelöst (vgl. dazu Abschnitt 6.4).

5. Die Struktur der Daten in ORACLE

In den vorangegangenen Abschnitten 1 bis 4 wurden die inhaltlichen Voraussetzungen sowie die lokalen Gegebenheiten (hard- und software- Ausstattung) für die Datenbankanwendung dargelegt. In den folgenden Abschnitten 5 bis 7 wird die Anwendungsentwicklung beschrieben. Im Abschnitt 5.1 werden die Kategorien und Werte, die in die Datenbank eingehen sollen, aufgelistet und die Beziehungen, die zwischen den Kategorien bestehen, angegeben. In 5.2 werden einige formale Grundlagen des Relationenmodells dargestellt und anhand von 5.1 veranschaulicht. In 5.3 werden wir die Datenstruktur, wie wir sie für die Datenbank bestimmt haben, vorstellen.

5.1. Kategorien, Werte und Beziehungen

Im Abschnitt 3 wurden grundlegende Aspekte für die Strukturierung der Daten des Projekts "Äußerungsaufbau" exemplarisch beschrieben. Als Ordnungsebenen waren Phrasen, Äußerungen, Design und Texte bestimmt worden. Des weiteren wurden pro Ordnungsebene einige der derzeit für die Projektfragestellung als relevant erachteten Segmente und Klassifikationskategorien mit den dazugehörigen Werten dargestellt. In einer ersten Annäherung sind als Daten die Analyseeinheiten, Kategorien und die Werte pro Kategorie gegeben. Wie diese im einzelnen bestimmt sind und welche Beziehungen zu berücksichtigen sind, wird nachfolgend dargestellt.

Für die Texte, mit denen das Projekt arbeitet, sollen die Bedingungen, unter denen sie erhoben wurden, zur Verfügung stehen, was durch die Ordnungsebene DESIGN erfaßt ist. Zu ihrer Charakterisierung werden überwiegend nicht-linguistische Kategorien herangezogen, aber auch linguistische wie etwa "Diskurstyp". Dies deshalb, weil die Informanten vor der Textproduktion u.a. beispielsweise dazu aufgefordert wurden, einen Sachverhalt zu beschreiben und somit der angestrebte Diskurstyp "Beschreibung" bereits gesetzt war. Insgesamt bildet die Design-Ebene die Merkmale eines Designs - hier verstanden als terminus technicus - ab, d.h. die Klassifikationskategorien sind die unabhängigen Variablen, die Werte einer Kategorie die Ausprägungen oder Stufen der unabhängigen Variablen.

Auf der Ordnungsebene ÄUßERUNGEN werden für die Analyse des Äußerungsaufbaus die Texte in Äußerungen segmentiert und diese anhand linguistischer Kategorien, die Äußerungen als Ganze betreffen, klassifiziert. Es sei hier vorweggenommen, daß in Hinblick auf die Datenbank das zu analysierende Element (die Äußerung) den gleichen Status hat wie die Werte einer Klassifikationskategorie.

Wie die Äußerungsebene ist auch die Ordnungsebene PHRASEN dadurch gekennzeichnet, daß die Äußerungen in Phrasen segmentiert und diese wiederum nach linguistischen Kriterien klassifiziert werden.

Die vierte Ordnungsebene - TEXTE - umfaßt keine Klassifikation im eigentlichen Sinne, es wird lediglich jedem Text der Informantenname zugeordnet. Prinzipiell wäre diese Zuordnung auf der Design-Ebene vorzunehmen, da es hierbei um die Darstellung von Stichproben und Messwiederholung geht (vgl. S. 11f.). Unter datenbanktechnischen Gesichtspunkten ist es allerdings sinnvoll, an der Ordnungsebene TEXTE festzuhalten.

Für die Ordnungsebenen wurden im einzelnen folgende Kategorien und Werte bestimmt[15]:

5.1.1. DESIGN [16]

(01) Diskurstyp

[DISKURSTYP]
{BESCH, ERZAE, INSTR}

Als Diskurstypen sind Beschreibung, Erzählung und Instruktion vorgesehen.

(02) Muttersprache des Informanten und Äußerungssprache

[SPRACHE]
{DD, EE, IE, ID, CD}

Die Werte kennzeichnen Deutsch (D), Englisch (E), Italienisch (I) oder Chinesisch (C) als Muttersprache (erster Buchstabe) und Äußerungssprache (zweiter Buchstabe).

(03) Gegenstand des Diskurses

[OBJEKT]
{BADESEE, CHAPLIN, GABELSTAPLER, PUPPENSTUBE_01, PUPPENSTUBE_02, PUPPENSTUBE_03, PUPPENSTUBE_04, STREET-SCENE, DORF}

Die Werte verweisen lediglich auf komplexe Settings, für die Texte erhoben wurden. Die detaillierten Beschreibungen der Settings werden nicht über die Datenbank zur Verfügung gestellt.

---

15) Die angegebenen Kategorien und Werte sind keineswegs erschöpfend, sondern diejenigen, die bisher in die Datenbankanwendung eingegangen sind.

16) Die in [ ] und { } angegebenen Bezeichnungen sind diejenigen, die in der Datenbank verwendet werden.

(04) Hörermerkmale

[HOERERFEAT]
{KIND, STUDENT, LERNER, KONFIDENT}

Hier ist festgehalten, an wen der Informant seinen Text qua Instruktion adressiert. Unerheblich ist dabei, ob der Adressat während der Textproduktion anwesend war oder ob der Informant sich den Adressaten lediglich vorstellen sollte. Dies gilt nicht für den Wert KONFIDENT. Ein Konfident ist während der Textproduktion durch den Informanten anwesend. Da es sich bei einem Konfidenten in der Regel um einen Projektmitarbeiter handelt, wurde darauf verzichtet, weiter zu differenzieren, etwa in der Weise, daß man dem Informanten den Konfidenten als Student oder Lerner vorstellt und dies in der Datenbank entsprechend berücksichtigt.

(05) Kognitionen des Hörer bezüglich des Diskursgegenstands während der Kommunikationsphase

[HOERERCOGN]
{OBJ, GRUNDRISS, GRUNDR+PARTS, PARTS, NOINFO}

Die Werte halten fest, ob der Adressat der Textproduktion über den Diskursgegenstand vollständig verfügt (OBJ), gegebenenfalls nur über Teile (GRUNDRISS, GRUNDR+PARTS, PARTS) oder keines von beidem (NOINFO).

(06) Wissen des Sprechers über die Aktivität(en) des Hörers auf dem Hintergrund der kommunizierten Information

[HOERERACTIV]
{EINRICHTEN, MENT_BILD, NACHBAUEN, OBJ_FINDEN, ZEICHNEN}

Über die Werte ist bestimmt, in welcher Weise der Informant vor der Textproduktion instruiert wurde. So soll z.B. in Hinblick auf die Puppenstuben der Adressat in der Lage sein, diese aufgrund der Informantenäußerungen einzurichten (EINRICHTEN) oder lediglich eine Vorstellung von einer Puppenstube und deren Einrichtung zu entwickeln (MENT_BILD) oder ein bestimmtes Objekt in einem Setting zu finden (OBJ_FINDEN). Desgleichen kann die Aktivität des Adressaten darin bestehen, ein Objekt NACHzuBAUEN, ein bestimmtes Setting zu ZEICHNEN, etc.

(07) Sprechermerkmale

[SPRECHERFEAT]
{B_SCHUELER, STUDENT, LERNER}

In gleicher Weise wie der Hörer charakterisiert wird, sind auch die Informanten zu beschreiben, wobei B_SCHUELER für Berufsschüler steht, und STUDENT und LERNER wohl keiner Erläuterung mehr bedürfen.

(08) Kognition des Diskursgegenstands durch den Sprecher

[SPRECHERCOGN]
{ICENTRY, ICLOFENTRY, ICOPPOFENTRY, ICROFENTRY, ICUNDEFINED, PCENTRY, PCLOFENTRY, PCOPPOFENTRY, PCROFENTRY, PCUNDEFINED}

Hier wird durch die Werte festgehalten, in welcher Weise der Diskursgegenstand durch den Informanten wahrgenommen wurde. Zunächst wird danach getrennt, ob er den Diskursgegenstand sieht, während er darüber spricht (IC) oder nachdem er ihn gesehen hat (PC). Des weiteren wird differenziert, aus welcher Position der Informant den Gegenstand wahrnimmt/wahrgenommen hat. Hierbei wird von den Objekten

PUPPENSTUBE und DORF ausgegangen, die durch einen Eingang charakterisiert sind. Erfolgt die Wahrnehmung von Eingang aus, wird der Wert ICENTRY oder PCENTRY vergeben. Der Eingang wird als Referenzpunkt für die übrigen Werte verwendet, d.h. IC/PC zusammen mit LOFENTRY bedeutet, daß der Gegenstand von der Seite links von Eingang wahrgenommen wird, OPPOFENTRY von der dem Eingang gegenüberliegenden Seite, ROFENTRY von der Seite rechts vom Eingang. Der Wert UNDEFINED charakterisiert alle übrigen Wahrnehmungspositionen.

(09) Sprecher/Hörerrelation während der Kommunikation

[SPRECHERCOMM]
{CANONICAL, SASH, SLOFH, SROFH, TELEFON}

Durch die Werte wird die Kommunikationssituation beschrieben, wobei CANONICAL bedeutet, daß Informant und Adressat einander gegenübersitzen, SASH bedeutet, daß sie nebeneinander sitzen, SLOFH besagt, daß der Informant links von Adressaten sitzt und SROFH entsprechend rechts vom Adressaten.

(10) spezifische Produktionsbedingungen

[PRODBEDING]
{OBJEKT, PLAN+MEMO, PARTS+MEMO, MEMORY}

Über diese Werte wird festgehalten, auf welche Information sich der Informant während der Textproduktion stützen kann. Er verfügt entweder über das OBJEKT oder einen Plan bzw. Teile des Objekts und muß die restliche Information aus dem Gedächtnis abrufen (PLAN+MEMO bzw. PARTS+MEMO), oder er muß die gesamte Information aus dem Gedächtnis abrufen (MEMORY).

Eine Erhebungsbedingung wird durch die Angabe eines Wertes pro Kategorie beschrieben.

So kennzeichnen z. B. "ERZAE, CD, CHAPLIN, KONFIDENT, NOINFO, MENT_BILD, STUDENT, PCUNDEFINED, CANONICAL, MEMORY" eine mögliche Erhebungsbedingung, in der von den Informanten eine Erzählung verlangt wird. Gegenstand der Erzählung ist eine Episode aus einem Film mit Chaplin. Die Informanten können lediglich auf Gedächtnisinhalte zurückgreifen. Sie sind Studenten, ihre Muttersprache ist Chinesisch, die Äußerungssprache Deutsch. In dieser Bedingung ist der Hörer ein Konfident. Informant und Konfident sitzen einander gegenüber, während der Informant erzählt. Die Informanten sind gehalten, davon auszugehen, daß der Hörer die Episode nicht kennt und ihm eine Vorstellung davon zu vermitteln ist. Da die Informanten einen Filmausschnitt gesehen haben, wird die Kognitionssituation als undefiniert bestimmt.

Für die Beziehungen zwischen den Kategorien gilt, daß formal prinzipiell vollständig kombiniert werden kann, d.h. ein Diskurstyp kann mehrere Objekte zum Gegenstand haben (Beschreibungen können vorgenommen werden für einen Gabelstapler, eine Puppenstube, etc.) oder umgekehrt ein Objekt kann Gegenstand mehrerer Diskurstypen sein (für einen Gabelstapler kann eine Beschreibung oder eine Instruktion produziert werden, etc.). Dies gilt auch für die übrigen Kategorien. Bestimmte Wertekombinationen kodieren die gleiche Information doppelt und bestimmte Werte einer Kategorie schließen bestimmte Werte aus einer anderen Kategorie aus, erfüllen aber eben deswegen Kontrollfunktionen innerhalb der Datenbank.

5.1.2. ÄUßERUNGEN

(11) wer ist der Sprecher der Äußerung — [SPRECHER] {I, K, P, V}

Während bei der Ordnungsebene DESIGN mit Sprecher immer der Informant gemeint ist, gilt für die Ebene ÄUßERUNGEN, daß eine in der Datenbank vorkommende Äußerung nicht notwendigerweise vom Informanten sein muß. Aus diesem Grund wird hier in der Kategorie SPRECHER unterschieden zwischen Informant (I), Konfident (K), Partner (P), ein in der Kommunikationssituation anwesender Hörer, der eben nicht Konfident ist, und nicht zuletzt dem Versuchsleiter (V).

(12) Wortlaut der Äußerung — [WORTLAUT] {WERTE: sind die als Äußerungen klassifizierten Textteile}

(13) Anzahl der pro Äußerung vorkommenden Wortformen — [WORTANZAHL] {1, 2,..., n}

(14) enthält die Äußerung ein Agens — [AGENS] {+, -}

Mit dieser Information wird angezeigt, ob etwa in einer Instruktionsäußerung die Person, die die ausgedrückte Handlung durchführen soll, explizit genannt wird oder nicht; so ist z.B. "das graue Teil schiebst du jetzt da drauf" durch (+) und "das graue Teil wird jetzt da drauf geschoben" durch (-) zu klassifizieren.

(15) an welcher Position steht das Finitum — [POSFINITUM] {E, Z, D, V, L, F}

Hier wird unterschieden zwischen Äußerungen, die keine finite Verbform enhalten (F) und solchen mit Erst- (E), Zweit- (Z), Dritt- (D), Viert- (V) und Letztstellung (L) des Finitums.

(16) um welche Illokution handelt es sich — [ILLOKUTION] {AFF, IMP, INT, SUB, SON}

Die Abkürzungen stehen für Affirmation, Imperation, Interrogation, Subjunktion und SON für nicht klassifizierbar. Illokution ist hier nicht im Sinne der Sprechakttheorie verstanden, sondern im Sinne einer Semantik von Satzmodalitäten.

(17) welche Diskursfunktion hat die Äußerung — [DISKURSFUNKTION] {H, N}

Mit dieser Information wird ausgesagt, ob eine Äußerung zum Hauptstrukturteil (H) des Textes gehört und damit also auf die Textfrage direkt bezogen ist oder zu einem Nebenstrukturteil (N); das beeinflußt die Topikalitäts- und Fokussiertheitsverteilung der Informationen der Äußerung.

(18) welche Funktion hat sie bezüglich des in ihr vorkommenden Gegenstands

[FUNKTION]
{INTRO, LOK, PROP, MANIP, INTRO+LOK, INTRO+PROP, INTRO+MANIP, INTRO+LOK+PROP, INTRO+LOK+MANIP, INTRO+LOK+PROP+MANIP, LOK+PROP, LOK+MANIP, LOK+PROP+MANIP, PROP+MANIP}

Über diese Werte wird bestimmt, ob durch die Äußerung ein Objekt eingeführt wird (INTRO), ob ein Objekt lokalisiert wird (LOK), ob einem Objekt eine Eigenschaft zugeschrieben wird (PROP) oder ob an einem Objekt oder mit einem Objekt eine Manipulation durchgeführt wird (MANIP). Desgleichen werden Kombinationen aus den vier Werten erfaßt, wobei die Reihenfolgen in einem kombinierten Wert nicht die Reihenfolge innerhalb der Äußerung widerspiegelt.

Eine Äußerung wird durch die Angaben eines Wertes pro Kategorie beschrieben. Der Wortlaut einer Äußerung ist als ein Wert der Kategorie WORTLAUT zu verstehen.

Die Beziehungen zwischen den Kategorien der Ordnungsebene ÄUßERUNGEN stellen sich minimal anders dar als bei DESIGN. Zur Erinnerung, dort war es so, daß ein Diskurstyp mehrere Objekte zum Gegenstand haben oder umgekehrt ein Objekt Gegenstand mehrerer Diskurstypen sein konnte und daß diese Art von Beziehung dort auch für die übrigen Kategorien galt. Bei den Beziehungen zwischen den Kategorien der Ordnungsebene ÄUßERUNGEN trifft dies beispielsweise nicht für die Beziehung zwischen WORTLAUT und WORTANZAHL zu. Hier verhält es sich vielmehr so, daß ein Wert aus der Kategorie WORTLAUT mit genau einem Wert aus der Kategorie WORTANZAHL in Beziehung steht. Ansonsten gilt aber ebenfalls, daß ein Wert aus einer Kategorie mit mehreren Werten aus einer anderen Kategorie in Beziehung steht.

5.1.3. PHRASEN

(19) der Wortlaut der Phrase wird festgehalten

[WORTLAUT_PHR]
{WERTE: sind die als Phrasen klassifizierten Äußerungsteile}

(20) welche syntaktische Funktion hat die Phrase

[SYNTFUNKT]
{SARG, PARG, SPR, PRPR, PADV, SADV, ATTR, NEG}

Die Werte bezeichnen syntaktische Funktionen - also das, was in der lateinischen Grammatik Subjekt, Prädikat, Attribut usw. genannt wird. Da die lateinische Grammatik für unsere Zwecke aber nicht genau genug ist, beschreiben wir die Struktur der Äußerung mit einer kategorial notierten Syntax. Die Funktionen werden - wie dabei üblich - als Beziehungen zwischen Konstituenten definiert. SARG ist die nominale Gruppe, die direkt von S dominiert wird, PARG die Objektgruppe, PADV das Prädikatsadverb. Die Bezeichnungen sind nicht mit den Definitionen identisch, sondern unter mnemotechnischen Gesichtspunkten ausgewählt.

(21) zu welcher syntaktischen Kategorie gehört sie [SYNTKATEG]
{N, NP, NK, KS, VFIN, V, VP, KOP+ADJ, PP, INFS, ADV,}

Über die Werte werden syntaktische Klasseneigenschaften erfaßt. Die Klassen sind im Prinzip distributionell definiert, wie in einer Phrasenstrukturgrammatik.

(22) in welcher Ausprägung liegt die Agentivität der Phrase vor [AGENTIV]
{A5, A4, A3, A2, A1, A0, P1, P2, P3, P4}

Hier wird angegeben, welche "Agentivitätsausprägung" eine nominale Phrase in Relation zum verbalen Knoten hat. Agentivität beschreiben wir im Sinne von Dowty (1989) als inhärente semantische Eigenschaften einer Verb-Argument-Beziehung. Sie läßt sich letztliche bestimmen als Art der Abhängigkeit des im Argument bezeichneten Referenten vom Ereignis, das das Verb bezeichnet, bzw. umgekehrt. A5 bis A0 bezeichnen Stufen der Abhängigkeit des Ereignisses vom Referenten, P1 bis P4 Abhängigkeiten des Referenten vom Ereignis.

(23) welchen Referenzbereich hat die Phrase [REFBEREICH]
{PERS/OBJ, ZEIT, RAUM, MOD, AKT/EREIGN}

Eine Äußerung ist die Verbalisierung einer Proposition, die ihrerseits eine Struktur von Referenten ist. Referenten können auf Orte, Zeiten, Personen und Objekte, Modalitätsausprägungen und Ereignisse bzw. Aktivitäten verweisen. Hier wird angegeben, was für die jeweilige Phrase zutrifft.

(24) welche semantischen Merkmale [SEMMERKMALE]
{PERS, BEL, SACHVERH, CAUS, ADVERS, GEGENSTDL, FINAL, INSTR, KONZESS, KOMP, KONSEK}

Es gibt Hinweise darauf, daß die Position, die eine Phrase in der Äußerung einnimmt, auch davon bestimmt wird, welchen Inhalt sie hat. Die in der Forschung bisher als relevant geltenden Merkmale werden mit dieser Information kodiert.

(25) welchen Kontextbezug [KONTEXTBEZUG]
{ERST, NEU, VERSCHIEB, ERH, WIEDER}

Hier wird angezeigt, welche Beziehung der in der Phrase bezeichnete Referent zu Phrasen in vorangehenden Äußerungen hat, ob er zum erstenmal erwähnt oder wiedererwähnt wird, ob an einen Bezugsreferenten angeknüpft wird usw.

(26) an welcher Position steht sie innerhalb der Äußerung [APOS]
{1, 2, 3,..., 1bis2, 2und4,....}

Diese Angabe bezeichnet die absolute Position der Phrase (d.h. all ihrer Wortformen) in der ganzen Wortformenkette der Äußerung. Die Position wird einfach kodiert, indem die Wortnummer(n) der zur Phrase gehörenden Wortform(en) angegeben wird (werden).

(27) welche Position nimmt sie relativ zum KOPF ein — [REIHENFOLGE]. {VOR, NACH, HEAD}

In der Bindungs- und Rektionstheorie (GB) wird bei syntaktischen Funktionen auch die des "Kopfes" einer Konstituente beschrieben; damit wird eine Struktureigenschaft des Satzes erfaßt, die auch Einfluß auf die relative Reihung der Phrasen in der Äußerung hat. Wir kodieren also für jede Phrase, ob sie in der gegebenen Äußerung VOR oder NACH dem "Kopf" steht oder selbst "Kopf" von anderen Phrasen in der Äußerung ist.

Für die Ordnungsebene Phrasen gilt, daß eine Phrase durch die Angabe eines Wertes pro Kategorie zu beschreiben ist.

Hier gelten die gleichen Beziehungen zwischen den Kategorien wie bei der Ordnungsebene DESIGN, d.h. ein Wert einer Kategorie kann mit mehreren Werten einer anderen Kategorie in Beziehung stehen.

Um Mißverständnisse zu vermeiden, sei der Unterschied zwischen "Beschreiben eines Wertes einer Kategorie durch einen Wert einer anderen Kategorie" und "ein Wert einer Kategorie steht mit mehreren Werten einer anderen Kategorie in Beziehung" hier nochmals erläutert: Eine Phrase (Wortlaut der Phrase = ein Wert der Kategorie WORTLAUT_PHR) kann beispielsweise in der Umgebung U1 als HEAD (Wert der Kategorie REIHENFOLGE) klassifiziert (beschrieben) werden, in einer anderen Umgebung U2 würde die gleiche Phrase (der gleiche Wortlaut) als VOR und in einer weiteren Umgebung U3 als NACH klassifiziert werden, aber eben in U1 nur als HEAD, etc. Die Beziehung, die hingegen zwischen dem Wert aus WORTLAUT_PHR und den Werten aus REIHENFOLGE besteht, ist eine "eins zu vielen"-Beziehung. Streng genommen handelt es sich zwischen beiden Kategorien um eine viele zu vielen-Beziehung, da ja auch gilt, daß ein Wert aus REIHENFOLGE mit mehreren Werten aus WORTLAUT_PHR in Beziehung steht.

5.1.4. TEXTE

Weiter oben (S. 11f.) wurde eine Ordnungsebene TEXTE mit Informationen über Stichproben und Messwiederholung postuliert, und auf Seite 18 wurde darauf verwiesen, daß diese Information zur DESIGN-Ebene gehört. Da aber aus datenbanktechnischen Gründen eine "Ebene" TEXTE notwendig ist, gehen wir hier bereits darauf ein. Die einzige "Kategorie" dieser Ebene ist der Informantenname. Dieser Kategorie ist als weitere Kategorie Textnummer hinzuzufügen. Beide zusammen bilden in angemessener Weise Messwiederholungen ab.

(28) Textnummer — [TNR] {1, 2, 3,..., n}

(29) Name des Informanten — [INAME] {$NAME_1$, $NAME_2$,..., $NAME_n$}

Jedem Text wird zunächst eine Textnummer zugordnet und dieser wiederum der Name des Informanten, der ihn produziert hat. Eine Textnummer wird durch genau einen Informantennamen charakterisiert

(beschrieben). Zwischen beiden Kategorien bestehen folgende Beziehungen: Ein Wert aus der Kategorie TNR steht mit genau einem Wert aus der Kategorie INAME in Beziehung und ein Wert aus INAME steht mit mehreren Werten aus TNR in Beziehung, wodurch die Messwiederholung gekennzeichnet ist.

5.2. Formale Grundlagen des Relationenmodells

Die Anwendung von Relationen im Zusammenhang mit Datenbankmanagementsystemen wird zum erstenmal von Codd (1969) beschrieben. Gut zwei Jahrzehnte später schreibt er: "The relational model...does not specify how a DBMS should be built, but it does specify what should be built, and for that it provides a precise specification. An important adjunct to precision is a sound theoretical foundation. The relational model is solidly based on two parts of mathematics: first order predicate logic and the theory of relations." (Codd, 1990, S. v). Im Laufe der Jahre hat er sein ursprüngliches Modell weiterentwickelt und die derzeit publizierte Version - RM/V2 - basiert auf 20 grundlegenden Gesetzen und 333 Merkmalen. Wir werden in 5.2 weder auf die Gesetze noch auf irgendeines der Merkmale explizit eingehen, sondern uns damit begnügen, einige wenige mathematische Grundlagen und Begrifflichkeiten darzustellen, die wir für ein allererstes Verstehen und Umgehen mit RDBMS und für die Anwendungsentwicklung als notwendig erachten.

a) Eine Menge M

ist eine Zusammenfassung wohlunterschiedener Objekte, d.h. verschiedener, eindeutig identifizierbarer Objekte. Danach läßt sich eine Kategorie als Menge bestimmen, mit den Werten als wohlunterschiedenen Objekten.

b) Funktionale Abhängigkeit (Typ 1 Assoziation; a —> b)

besteht zwischen zwei Mengen M_1 und M_2, wenn es eine Funktion gibt, die jedem Elemente der Menge M_1 genau ein Element der Menge M_2 zuordnet.

Eine Typ 1 Assoziation besteht zwischen WORTLAUT und WORTANZAHL, denn jedem Elemente aus WORTLAUT (jeder Äußerung) ist genau ein Element aus WORTANZAHL (1, 2, ..., n) zugeordnet. Desgleichen besteht funktionale Abhängigkeit zwischen TNR und INAME; jedem Wert aus TNR ist genau ein Wert aus INAME zugeordnet.

c) Das kartesische Produkt

zweier Mengen M_1 und M_2 ist die Menge $M_1 \times M_2$, die alle Paare (a,b) (2er-Tupel) enthält, für die gilt: $a \in M_1$ und $b \in M_2$; M_1 und M_2 sind beliebige Mengen, können im besonderen auch gleich sein.

$$M_1 \times M_2 := \{(a,b) / a \in M_1 \wedge b \in M_2\}$$

Das kartesische Produkt der Mengen SPRECHER und ILLOKUTION ergibt eine Menge, die als Elemente die Tupeln (I, AFF),..., (V, SON) enthält:

SPRECHER x ILLOKUTION := {(I, AFF), (I, IMP), (I, INT), (I, SUB), (I, SON),
(K, AFF), (K, IMP), (K, INT), (K, SUB), (K, SON),
(P, AFF), (P, IMP), (P, INT), (P, SUB), (P, SON),
(V, AFF), (V, IMP), (V, INT), (V, SUB), (V, SON)}

Das kartesische Produkt kann auch von mehreren Mengen $M_1....M_n$ gebildet werden:

$$M_1 \times M_2 \times ... \times M_n = \{(a_1,a_2,...,a_n) / a_i \in M_i \quad \forall i = 1...n\}$$

Nimmt man zu SPRECHER und ILLOKUTION noch AGENS hinzu, dann umfaßt das kartesische Produkt als Elemente die Tupeln {(I,AFF,+),..., (V, SON, -)}

d. Eine Relation R

zweier Mengen M_1 und M_2 ist eine Teilmenge des kartesischen Produkts $M_1 \times M_2$:

$$R(M_1, M_2) \subseteq M_1 \times M_2$$

In ihr liegen alle die Elemente (a,b) mit $a \in M_1$, $b \in M_2$, für die aRb eine wahre Aussage ist ("a steht zu b in der Relation R"). Ebenso, wie in einer Menge ein Element nur einmal vorkommen kann, kann in einer Relation, die ja ebenfalls eine Menge ist, ein Element, d.h. ein geordnetes Tupel, nur einmal vorkommen.

Gegeben die Bedingung, daß der Konfident sich während der Texterhebung nicht verbal äußern darf, ergibt sich zwischen SPRECHER und ILLOKUTION die Relation:

R(SPRECHER, ILLOKUTION) := {(I, AFF), (I, IMP), (I, INT),
(I, SUB), (I, SON), (P, AFF),
(P, IMP), (P, INT), (P, SUB),
(P, SON), (V, AFF), (V, IMP),
(V, INT), (V, SUB), (V, SON),}

Der Begriff der Relation kann erweitert werden zur Relation n-ten Grades über den Mengen $M_1, M_2,..., M_n$:

$$R(M_1, M_2,..., M_n) \subseteq M_1 \times M_2 \times ... \times M_n$$

Jede n-stellige Relation $R(M_1, M_2,..., M_n)$ besteht also aus einer Menge von Elementen $(a_1,a_2,...,a_n)$ für die gilt:

$$a_i \in M_i \; \forall i = 1...n, \text{ und } R(a_1,a_2,...,a_n) \text{ ist eine wahre Aussage.}$$

Die Menge M_1 bildet den Defintionsbereich der Relation. Die Mengen M_2, M_3,..., M_n sind die Attribute mit den Attributwerten als Elementen.

Dem hier dargelegten Relationenbegriff der Mathematik sei die Definition von Codd (1990, S. 2) hinzugefügt, da hierdurch zugleich der Begriff der Domäne eingeführt wird: "...R is a subset of the Cartesian product S1 x S2 x ... x S*n*....Relation R is said to be of degree *n*. Each of the sets S1, S2,..., S*n* on wich one or more relations are defined is called a *domain*." Relationen (R) können auch als Tabellen verstanden werden, die dann folgenden Eigenschaften haben: "each row represents a tuple of R; the ordering of rows is immaterial; all rows are distinct from one another in content." (Codd, 1990, S. 2). Zwischen Relationen und Tabellen besteht jedoch wenigstens ein gravierender Unterschied. In Tabellen sind generell zwei oder mehr Zeilen mit identischem Inhalt zulässig, in Relationen jedoch ausgeschlossen. Wir werden dennoch beide Begriffe synonym verwenden und zwar in der Bedeutung von Relation.

5.3. Datenstrukturierung für die Datenbank

Das relationale Modell ist mit drei Merkmalen von Daten befaßt: der Struktur, der Integrität und der Manipulation der Daten. Wenngleich die Struktur der Daten den Schwerpunkt dieses Abschnitts bildet, werden im Zusammenhang damit auch immer wieder Aspekte der Datenintegrität und -manipulation behandelt werden. Die Bestimmung der Datenstruktur kann auf recht unterschiedliche Weise erfolgen, und diese verschiedenen Ansätze werden kontrovers diskutiert (vgl. u.a. Codd, 1990, S. 6f. und Kap. 30; Date, 1990, Kap. 22). Bei der Strukturierung der Projektdaten haben wir uns keinem der Ansätze ausschließlich verschrieben, folgen weitgehend Codd (1990) und Date (1990), nutzen aber hin und wieder Elemente des entity/relationship-Ansatzes, der von Chen (1976) vorgeschlagen wurde und an dem sich u.a. Vetter (1990), wenn auch nicht explizit, orientiert. Daß dieser Ansatz, wie Codd (1990, S. 7) bemängelt, nur strukturelle Aspekte beschreibt, kommt uns dabei entgegen.

Bei der Strukturierung der Daten müssen wir im Auge behalten, (a) mit welcher Begrifflichkeit zu arbeiten ist, (b) welche Eigenschaften Relationen haben, und (c) welche Arten von Relationen beschrieben werden.

(a) Date (1990, S. 250) legt als Datenstrukturterminologie folgende Begriffe fest, von denen einige bereits formal definiert wurden und die übrigen, soweit wir sie verwenden, dort paraphrasiert werden:

"Formal relational term	Informal equivalent
relation	table
tuple	row or record
cardinality	number of rows
attribute	column or field
degree	number of colums
primary key	unique identifier
domain	pool of legal values"

(b) Relationen können aufgrund ihrer mathematischen Definition durch vier Eigenschaften gekennzeichnet werden:

"There are no duplicate tuples;
Tuples are unordered (top to bottom);
Attributes are unordered (left to right);
All attribute values are atomic." (Date, 1990, S. 261)

(c) Innerhalb eines relationalen Systems unterscheidet man verschiedene Arten von Relationen (vgl. Date, 1990, S. 265f.). Unser Ziel ist es, ausgehend von der Gesamtheit der Daten, diejenigen "base relations" zu bestimmen, die den Grundstock der Datenbank bilden und u.a. erlauben, virtuelle Relationen (views) zu erzeugen, seien diese Ausschnitte aus einer Relation oder Kombinationen aus Teilen von verschiedenen Relationen.

5.3.1. Die Gesamtheit der Projektdaten

Betrachten wir also zunächst die Gesamtheit der Daten des Projekts als die Relation A1-DATEN (DISKURSTYP,..., INAME) anhand dreier Beispieltexte, die von zwei Sprechern BEN und TOM stammen. Ihre Muttersprache ist Englisch, die Äußerungssprache Deutsch. Tom wurde angehalten, eine Beschreibung vorzunehmen, Ben eine Beschreibung (1) und später eine Erzählung (2). Die Texte haben folgenden Wortlaut:

TOM: ich ging
BEN (1): der mann kommt / hanna geht
BEN (2): läuft das kind

Die Segmentierung der Texte ergibt vier Äußerungen, die Segmentierung der Äußerungen führt zu acht Phrasen. In die Relation A1-DATEN sind somit 8 Tupeln einzugeben. Die wichtigsten Aspekte der Datenstrukturierung lassen sich anhand dieser Äußerungen und der Attribute DISKURSTYP (DT), SPRACHE (MÄ), WORTLAUT, W-ANZAHL, WORTLAUT_PHR, SYNTKATEG, TNR und INAME darstellen. Eine entsprechende Relation könnte z.B ohne weiteres so aussehen:

A1 TNR	INAME	MÄ	DT	WORTLAUT	W-ANZAHL	WORTLAUT-PHR	SYNTKATEG
1	TOM	ED	BESCH	ich ging	2	ich	NP
1	TOM	ED	BESCH	ich ging	2	ging	VP
2	BEN	ED	BESCH	hanna geht	2	Hanna	NP
2	BEN	ED	BESCH	hanna geht	2	geht	VP
3	BEN	ED	ERZAE	läuft das kind	3	läuft	VP
3	BEN	ED	ERZAE	läuft das kind	3	das kind	NP
2	BEN	ED	BESCH	der mann kommt	3	kommt	VP
2	BEN	ED	BESCH	der mann kommt	3	der mann	NP

A1 weist alle Eigenschaften einer Relation auf, d.h. es kommen keine identischen Tupeln vor; die Tupeln sind nicht sequentiell geordnet, d.h. die Reihenfolge der Tupeln richtet sich z.B. nicht nach der Ordnung der

Äußerungen im Text (die erste Äußerung des zweiten Texts steht im vorletzten und letzten Tupel der Tabelle, was unter inhaltlichen Gesichtspunkte natürlich nicht gerade sinnvoll ist); die Reihenfolge der Attribute ist ebenso willkürlich (man hätte auch mit SYNTKATEG anfangen können) und die Attributwerte sind "atomic", d.h. an keinem Zeilen/Spalten-Schnittpunkt kommen mehrere Werte vor, wie es der Fall wäre, wenn man die Äußerung "ich ging" so darstellen würde:

A2	TNR	INAME	MÄ	DT	WORTLAUT	W-ANZAHL	WORTLAUT-PHR	SYNTKATEG
	1	TOM	ED	BESCH	ich ging	2	ich / ging	NP / VP

Ganz offensichtlich ist die Relation A1 unter inhaltlichen Gesichtpunkten, wegen der mangelnden Reihenfolgeinformation, nicht brauchbar, und in Hinblick auf die Datenbankanwendung weist sie erhebliche Mängel auf. Zwar kann man in dieser Relation die einzelnen Tupeln über die Werte des Attributs WORTLAUT_PHR unterscheiden bzw. identifizieren, womit WORTLAUT_PHR als Primärschlüssel fungieren könnte, generell gilt aber, daß ein und derselbe Wortlaut einer Phrase mehrmals vorkommen kann und auch vorkommen wird. Entscheidend ist jedoch, daß eine Relation wenigstens ein Attribut oder eine Attributkombination aufweist, das oder die als Primärschlüssel dient. Prinzipiell könnte man die Tupeln durchnumerieren, d.h. man orientiert sich an der Anzahl der Phrasen. Die Rekonstruktion der Reihenfolge der Äußerungen innerhalb eines Textes würde dann zu einem mühseligen und ausgesprochen fehleranfälligen Unterfangen.

Im vorliegenden Fall bietet es sich daher eher an, als weitere Attribute wenigstens eine Äußerungsnummer ANR und eine Phrasennummer PNR einzusetzen, die, kombiniert mit TNR, den Primärschlüssel bilden. Gleichzeitig läßt sich mit Hilfe des Attributs ANR auch die Reihenfolge der Äußerungen im Text berücksichtigen und das am einfachsten, wenn man die Äußerungen pro Text beginnend mit 1 durchnumeriert. Das gleiche Verfahren erweist sich auch als sinnvoll für die Phrasen pro Äußerung. Die entsprechende Relation sähe dann wie folgt aus:

A3	TNR	ANR	PNR	INAME	MÄ	DT	WORTLAUT	W-ANZAHL	WORTLAUT_PHR	SYNTKATEG
	1	1	1	TOM	ED	BESCH	ich ging	2	ich	NP
	1	1	2	TOM	ED	BESCH	ich ging	2	ging	VP
	2	2	1	BEN	ED	BESCH	hanna geht	2	Hanna	NP
	2	2	1	BEN	ED	BESCH	hanna geht	2	geht	VP
	3	1	1	BEN	ED	ERZAE	läuft das kind	3	läuft	VP
	3	1	2	BEN	ED	ERZAE	läuft das kind	3	das kind	NP
	2	1	2	BEN	ED	BESCH	der mann kommt	3	kommt	VP
	2	1	1	BEN	ED	BESCH	der mann kommt	3	der mann	NP

Jeder Wert des Primärschlüssels (die Kombination TNR,ANR,PNR) kommt genau einmal vor, identifiziert also genau ein Tupel. Zudem besteht die Möglichkeit, den Primärschlüssel oder Teile davon als Sortierschlüssel zu verwenden, um z.B. die Reihenfolge der Äußerungen oder Phrasen des Textes mit TNR = 2 zu repräsentieren.

Man könnte sich mit dieser Relation begnügen bzw. mit A1-DATEN, erweitert um die Attribute ANR und PNR. Die Relationen, ob mit den Attributen ANR und PNR oder ohne sie, sind aber zum einen redundant, d.h. identische Information ist mehrfach festgehalten, so z.B. viermal, daß BEN eine Erzählung produzieren sollte und daß seine Muttersprache Englisch und die Äußerungssprache Deutsch ist; zum anderen enthalten die Relationen Informationen über durchaus unterschiedliche Sachverhalte, eben über die Untersuchungsbedingungen, Informanten, Äußerungen und Phrasen. Beides führt zu erheblichen Problemen (Speicheranomalien) bei den Speicheroperationen INSERT, DELETE und UPDATE. Um welche Probleme es sich dabei handeln kann, sei durch jeweils ein Beispiel veranschaulicht, wobei INSERT- und DELETE-Manipulationen als zwei Seiten einer Medaille betrachtet werden können. Bei diesen beiden Operationen spielt der Primärschlüssel eine entscheidende Rolle. Er ist durch zwei Eigenschaften gekennzeichnet:

"1. *Uniqueness:*
At any time, no two tuples of *R* have the same value for *K*.

2. *Minimality:*
If *K* is composite, then no component of *K* can be eliminated without destroying the uniqueness property."
(Date, 1990, S. 277)

Primärschlüssel und Fremdschlüssel, auf den wir weiter unten eingehen werden, tragen in erheblichem Maße dazu bei, die Integrität einer Datenbank zu gewährleisten. Unter Integrität versteht man, daß Sachverhalte, Ereignisse, etc., die keine Entsprechung in der realen Welt haben, nicht als Information in die Datenbank eingehen dürfen. Gibt man z.B. lediglich eine Phrase und die entsprechende PNR ein, nicht aber die Äußerung, aus der sie stammt, den Informantennamen und die TNR usw., dann verzichtet man auf diese Information, die ja in der Realität existiert. Mit Hilfe der ersten Integritätsregel (*entity integrity rule*) soll verhindert werden, daß derartige und andere, der Realität wiedersprechende Informationen, in die Datenbankt eingehen können; Die Regel lautet:

"*No component of the primary key of the base relation is allowed to accept nulls.*"[17] (Date, 1990, S. 279).

Unter diesen Voraussetzungen führt die Operation INSERT zu folgendem Problem in der Relation A3: Man kann keine DESIGN-Information (z.B. Chinesisch/Deutsch und Instruktion) in die Relation einfügen, solange es nicht wenigstens eine Phrase aus einer Äußerung aus einem Text gibt, der eben unter den einzugebenden Bedingungen erhoben wurde, weil der Primärschlüssel nicht nur nicht vollständig ist, sondern bei allen drei Schlüsselattributen "Null" (im Sinne von "information is missing") als Werte nehmen müßte. Nun geht aber üblicherweise die Information über eine Untersuchungsbedingung allen weiteren Informationen voraus, existiert also in der Realität, bevor die Texte etc. vorliegen.

---

17) By "null" here, we mean (as usual) that information is missing for some reason - e.g., the property does not apply, or the value is unknown (etc.)." (Date, 1990, S. 279). Date diskutiert im Anschluß an diese Regel und im Kapitel 15 die SQL-Lösung des "Null"-Problems und empfiehlt die Attribute einer Relation durchgängig als "NOT-NULL" zu definieren, es sei denn, man habe eine hieb- und stichfeste Begründung, auf die Definition zu verzichten.

Das Problem ist weniger vertrackt und wird deshalb noch deutlicher, wenn man die Operation DELETE betrachtet. Geht man davon aus, daß es einen Text gibt, der aus genau einer Phrase besteht, z.B. "geh" (Imperativ), dann ist die dazugehörige Information in einem Tupel gespeichert. Soll die Phrase und nur diese aus der Datenbank entfernt werden, wird damit zugleich die Information, daß es sich um die Äußerung "geh" aus dem Text XYZ, erhoben unter den Bedingungen ABC handelt, entfernt.

Die dritte Operation, UPDATE, führt in der Relation A3 ebenfalls zu Speicherproblemen und zwar im Zusammenhang mit Wiederholungsgruppen (repeating groups) wie z.B. viermal "BEN, ED, BESCH". Wenn der Informant nicht BEN sondern JOHN war, dann muß die Datenbank nach jedem Tupel, das TNR = 2 mit BEN verbindet, suchen und BEN in JOHN ändern. Werden nicht alle BENs für TNR = 2 in JOHNs geändert, befindet sich die Datenbank in einem inkosistenten Zustand: Sie enthält nun als Information, daß an einem Text zwei Informanten beteiligt sind.

Zu den bisher aufgezeigten Problemen mit den Relationen A3 bzw. A1-DATEN, die man auch als Universalrelationen bezeichnet, weil sie u.a. die gesamte Information enthalten, kommen noch weitere hinzu, wovon wenigstens eines für die Projektarbeit virulent ist: "loss of adaptability to changes in the kinds of information stored in the database" (Codd, 1990, S. 42)[18], virulent deshalb, weil damit zu rechnen ist, daß solche Änderungen vorkommen werden (vgl. auch die Ausführungen S. 15).

Die Alternative zum Arbeiten mit einer Universalrelation besteht in ihrer Zerlegung in mehrere Relationen. Diese kann nach höchst unterschiedlichen Kriterien erfolgen, u.a. durch die Bestimmung der verschiedenen Entitäten, die in der Universalrelation enthalten sind und die Erfassung der jeweiligen Entität durch eine eigene Relation; d.h. aber nicht, daß durch solche Relationen *per se* die Integrität der Datenbank gewährleistet ist oder Speicheranomalien nicht mehr vorkommen können. Eine andere Möglichkeit, die Universalrelation in mehrere kleinere Relationen zu überführen, bietet die "Normalisierung", ein Verfahren zur Zerlegung von Relationen ohne Informationsverlust ("non loss decomposition" (Date, 1990, S. 557)), d.h. die Relationen, die aus der Zerlegung resultieren, müssen insgesamt mindestens die gleiche Information liefern wie die Ausgangsrelation. Die Relationenzerlegung über die Normalisierung dient vor allen Dingen dazu, den möglichen Speicheranomalien zu begegnen. Beide hier genannten Möglichkeiten schließen einander nicht aus.

Wir werden nachfolgend nicht mehr mit den Relationen A3 bzw. A1-DATEN befaßt sein, sondern mit Relationen für die Entitäten, die wir definiert haben. Für diese werden zunächst die Primär- und Fremdschlüssel bestimmt, und im Anschluß daran werden wir prüfen, in welcher Normalform sie sich befinden, um sie gegebenenfalls weiter zu zerlegen.

---

18) Die Probleme, die mit Universalrelationen einhergehen, werden ausführlich von Codd (1990, S. 40f. und Kap. 30) und Date (1990, S. 714f.) diskutiert.

5.3.2. Relationen, Primär- und Fremdschlüssel

Die Beschreibung der Daten in den Abschnitten 3 und 5.1 legt nahe, DESIGN, TEXTE, ÄUßERUNGEN und PHRASEN als Entitäten anzunehmen und die Kategorien als Attribute, über deren Werte die Entitäten beschrieben werden. Somit lassen sich zunächst vier Relationen bestimmen, die als DESIGN, TEXTE, TALK[19] und PHRASEN bezeichnet werden sollen. Die in 5.1 aufgelisteten Kategorien bilden die Attribute der Relationen. Zur Identifikation der Tupeln werden als weitere Attribute DNR (Designnummer), ANR (Äußerungsnummer) und PNR (Phrasennummer) eingeführt. Die Primärschlüssel DNR in DESIGN und TNR in TEXTE sind einfach und werden fortlaufend vergeben. TALK enthält den zusammengesetzten Primärschlüssel TNR,ANR, wobei die Werte für ANR pro Text, beginnend mit "1" vergeben werden. TNR,ANR,PNR ist der zusammengesetzte Primärschlüssel von PHRASEN. Hier wird PNR pro Äußerung, beginnend mit "1" vergeben.

1. DESIGN (DNR, DISKURSTYP,..., PRODBEDING)
2. TEXTE (TNR, INAME)
3. TALK (TNR,ANR, SPRECHER,..., FUNKTION) und
4. PHRASEN (TNR,ANR,PNR, WORTLAUT_PHR,..., REIHENFOLGE)

Mit diesen Relationen ist es z.B. ohne weiteres möglich, Informationen über einzelne Sachverhalte (die jeweiligen Entitäten betreffend) einzugeben, was bei A3 nicht der Fall war. Allerdings erlauben diese Relationen, im Gegensatz zu A3, noch nicht, die vollständige Information abzurufen. Dazu ist die Verknüpfung der Relationen notwendig. Sie wird über einen Fremdschlüssel geleistet. Unter einem Fremdschlüssel versteht man ein Attribut (eine Kombination von Attributen) einer Relation, dessen Werte mit den Werten des Primäschlüssels in einer anderen Relation übereinstimmt müssen, d.h. beide Schlüssel beziehen ihre Werte aus derselben Domäne[20]. Ein Attribut (eine Kombination von Attributen) ist ein Fremdschlüssel (FK), wenn es die folgenden Eigenschaften aufweist:

"1. Each value of FK is either wholly null or wholly non null.(...)
2. There exists a base relation R1 with primary key PK such that each non null value of FK is identical to the value of PK in some tuple of R1."
(Date, 1990, S. 282)

---

19) Der Zeichensatz, der von ORACLE zur Verfuegung gestellt wird, enthaelt keine Umlaute etc.; "AEUSSERUNGEN" auch nicht, "Äußerungen" dagegen schon. TALK ist einfach leichter zu handhaben. Uebrigens wurden die Vertreter von ORACLE auf der ORACLE-Anwendertagung im November 1990 gefragt, wann sie gedaechten, den Zeichensatz zu erweitern. Die Frage rief bei den Tagungsteilnehmern (Anwendern) Heiterkeit hervor, bei den uebrigen (Repraesentanten der Firma ORACLE) betretene Gesichter und die Auskunft, dass man bemueht sei...

20) Da wir hier nur numerische Werte vergeben, scheint diese Feststellung zunächst trivial, ist es aber deswegen nicht, weil Schlüsselattribute genausogut nicht-numerische Werte haben konnen. Wenn z.B. der Primärschlüssel einer Relation R1 seine Werte aus der Domäne {A, B, C} bezieht, dann muß auch der Fremdschlüssel in einer Relation R2 seine Werte aus dieser Domäne beziehen, damit die beiden Relationen sinnvoll miteinander verknüpft werden können.

DESIGN	PK DNR	(01).....	(10)
	1	ERZAE	MEMO
	2	INSTR	MEMO
	3	BESCH	MEMO
	4	BESCH	PARTS
	5	INSTR	PARTS

TEXTE	PK TNR	FK DNR	INAME
	1	1	A
	2	1	B
	3	2	A
	4	2	C
	5	2	B

Die Relation DESIGN hat den Primärschlüssel DNR und die Attribute (01) bis (10), TEXTE hat entsprechend TNR als Primärschlüssel und (INAME) als Attribut, und als weiteres Attribut wird DNR als Fremdschlüssel mitgeführt. Über den Fremdschlüsselwert wird aus DESIGN die gewünschte Information bereitgestellt. Für die Relation TALK gilt, daß die Kombination TNR,ANR als Primärschlüssel fungiert und TNR allein als Fremdschlüssel. In der Relation PHRASEN bilden TNR,ANR,PNR den Primärschlüssel, die Kombination TNR,ANR dagegen den Fremdschlüssel. Während also die Primärschlüssel einer Relation die eindeutige Identifikation von Entitäten ermöglichen, referieren die Fremdschlüssel auf Entitäten.

Im Zusammenhang mit dem Primärschlüssel haben wir bereits eine Integritätsregel behandelt. Mit dem Konzept des Fremdschlüssels besteht eine zweite Integritätsregel (*referential integrity rule*), die besagt:

"*The database must not contain any unmatched foreign key.*" (Date, 1990, S. 284).

Wird in TEXTE etwa das Tupel (9, 17, X) eingefügt, dann verstößt dies gegen die Regel, da es in DESIGN keine Entität (Erhebungsbedingung) gibt, auf die mit "17" referiert werden könnte. Streng genommen darf nach dieser Regel ein Fremdschlüssel auch keine NULL-Werte annehmen. Unter welchen Umständen NULL-Werte bei Fremdschlüsseln zulässig und wie sie zu handhaben sind, wird von Codd (1990) im Zusammenhang mit den Ausführungen zur Differenzierung von "missing information" und bei Date (1990, S. 285ff.) im Abschnitt **"FOREIGN KEY RULES"** beschrieben. Läßt man NULL-Werte zu, dann ist das Einfügen des Tupels (9,..., X) möglich, wodurch man dann über mehr Information verfügt, als etwa die Relation A3 zu bieten vermag, nämlich daß es bereits einen Informanten gibt, der einen Text produzieren wird, wobei noch offen ist unter welchen Untersuchungsbedingungen. DNR ist übrigens der einzige Fremdschlüssel der als Wert "Null" (im Sinne von "missing information") annehmen kann. TNR in TALK und TNR,ANR in PHRASEN müssen einen Wert erhalten, da sie Bestandteile des Primärschlüssels sind. In einer Datenbank, die diese vier Relationen enthält ist es aber ohne weiteres möglich, Design-Information bereits dann einzugeben, wenn die Planung einer Untersuchung abgeschlossen ist, was ja in A3 nicht möglich war. Hier nun noch einmal die Relationen:

1. DESIGN (DNR, DISKURSTYP,.:., PRODBEDING)
2. TEXTE (DNR, TNR, INAME)
3. TALK (TNR,ANR, SPRECHER,..., FUNKTION) und
4. PHRASEN (TNR,ANR,PNR, WORTLAUT_PHR,..., REIHENFOLGE)

5.3.3. Weitere Zerlegung über die Normalisierung?

Der Normalisierungsprozeß besteht aus mehreren Hauptschritten, die als erste, zweite, dritte, vierte und fünfte Normalform (1NF, 2NF, usw.) bezeichnet werden[21]. Die Normalisierung der Relation is kein "Muß", und es mag gute Gründe dafür geben, in einer Anwendung nicht alle Relationen in 2NF oder höher zu implementieren, und selbst wenn Vetter (1990, Kap. 4) die Auffassung vertritt, zumindest für die Implementierung, Relationen in 3NF anzustreben, so ist auch dies nur eine Empfehlung.

Betrachtet man die beiden Extreme 1NF und 5NF, so kann man festhalten, daß 1NF ein "Muß" ist (die Definition einer Relation entspricht 1NF), und für 5NF stellt Vetter (1990, S. 200) fest: "...wie in der Praxis ausfindig zu machen ist, ob eine Relation die 5NF verletzt, ist selbst den Erfindern besagter Normalform unklar.", und er schließt seine Ausführungen über die Normalisierung mit einem Zitat aus Date (1981) ab: "It is tempting to suggest that such relations are pathological cases and likely to be rare in practice." Date (1990, S. 529) beurteilt das Verfahren wie folgt "...database design can be an extremely complex task (...). Normalization theory is a useful aid in this process, but it is not a panacea; anyone designing a database is certainly advised to be familiar with the basic techniques of normalization but we do not suggest that the design should necessarily be based on normalization principles alone."

Hilfreich ist das Verfahren während der Konzeptualisierung schon allein deshalb, weil seine Anwendung Klarheit über die Bedeutung der Daten und die Beziehungen zwischen ihnen erfordert, und durch die Anwendung des Verfahrens jene Datenausschnitte aufgedeckt werden, bei denen diese Klarheit nur vermeintlich bestand. Beurteilen wir nun jede der Relationen dahingehend, in welcher Normalform sie sich befindet, und ob es geraten ist, die eine oder andere weiter zu zerlegen. Hierbei werden wir uns auf 1NF bis 3NF beschränken.

Durch die ersten drei Normalformen werden folgende Bedingungen gesetzt:

1. Normalform (1NF)
"Eine Relation ... mit dem einfachen Schlüssel S oder ... dem zusammengesetzten Schlüssel S1,S2 ist in 1NF, wenn alle nicht dem Schlüssel angehörenden Attribute vom Schlüssel funktional abhängig sind." (Vetter, 1990; S. 159)

2. Normalform (2NF)
"...eine 2NF-Relation ist dadurch gekennzeichnet, daß jedes nicht dem Schlüssel angehörende Attribut funktional abhängig ist vom Gesamtschlüssel (1NF-Kriterium), nicht aber von einzelnen Schlüsselteilen." (Vetter, 1990; S. 165).

---

21) Date (1990) stellt im Kapitel 21 neben 1NF bis 5NF noch weitere Normalformen vor, wie z.B. die "Boyce/Codd-NF", die eine Erweiterung von 3NF darstellt, aber nicht gleich 4NF ist.

3. Normalform (3NF)
"...eine 3NF-Relation ist dadurch gekennzeichnet, daß jedes nicht dem Schlüssel angehörende Attribut funktional abhängig ist vom Gesamtschlüssel (1NF-Kriterium), nicht aber von einzelnen Schlüsselteilen (2NF-Kriterium). Ferner sind keine funktionalen Abhängigkeiten zwischen Attributen erlaubt, die nicht als Schlüsselkandidaten in Frage kommen." (Vetter, 1990; S. 179)

Daß jede der vier Relationen in 1NF ist, liegt auf der Hand. Ob die Relationen TALK und PHRASEN, deren Primärschlüssel eine Attributkombination ist, das 2NF-Kriterium erfüllen, und wenn ja, ob alle vier Relationen das 3NF-Kriterium erfüllen, bleibt abzuwarten.

Betrachten wir zunächst TALK. Hier ist zu prüfen, ob es Attribute gibt, die von TNR allein oder von ANR allein funktional abhängig sind[22]. Die Attribute von TALK, die nicht dem Schlüssel angehören, sind: SPRECHER, WORTLAUT, WORTANZAHL, AGENS, POSFINITUM, ILLOKUTION, DISKURSFUNKTION und FUNKTION. Der nachfolgende Tabellenausschnitt[23] zeigt deutlich, daß z.B zwischen TNR und SPRECHER, zwischen TNR und WORTLAUT usw. keine funktionale Abhängigkeit besteht. Dem Wert TNR = 2 werden die beiden Werte I und K aus SPRECHER zugeordnet, usw. Auch für die übrigen Nichtschlüsselattribute der Relation TALK gilt, daß sie nicht von TNR funktional abhängig sind.

TALK	TNR	ANR	SPRECHER	WORTLAUT	WORTANZAHL	ILLOKUTION
	1	1	I	ich ging	2	AFF
	2	1	I	der mann kommt	3	AFF
	2	2	I	hanna geht	2	AFF
	2	3	K	stimmt das	2	INT
	3	1	I	läuft das kind	3	INT

Für ANR und die Nichtschlüsselattribute gilt ebenfalls, daß keine funktionale Abhängigkeit zwischen beiden besteht. Wir können also festhalten, daß die Relation das 2NF Kriterium erfüllt.

Wenden wir uns nun der Relation PHRASEN zu, die die Attribute WORTLAUT_PHR, SYNTFUNKT, SYNTKATEG, AGENTIV, REFBEREICH, SEMMERKMALE, KONTEXTBEZUG, APOS und REIHENFOLGE hat. Um es kurz zu machen, sie erfüllt das 2NF Kriterium ebenfalls, was in dem Tabellenausschnitt deutlich wird. Was hier für TNR und die Attribute WORTLAUT_PHR und SYNTKATEG, bzw. ANR, bzw. PNR und die beiden Nichtschlüsselattribute gilt, trifft auch für die übrigen Attribute zu.

---

22) MEMO: Funktionale Abhängigkeit besteht zwischen zwei Mengen M_1 und M_2 wenn es eine Funktion gibt, die jedem Element der Menge M_1 genau ein Element der Menge M_2 zuordnet.

23) Wir benutzen die Tabellen, um die bestehenden Abhängigkeiten zu veranschaulichen und nicht, um sie daraus herzuleiten. Das Aufzeigen von Abhängigkeiten ist ein genuin semantischer Prozeß, der der Formalisierung vorausgehen muß (vgl. Abschnitt 5.1).

PHRASEN	TNR,ANR,PNR			WORTLAUT_PHR	SYNTKATEG
	1	1	1	ich	NP
	1	1	2	ging	VP
	2	1	1	der mann	NP
	2	1	2	kommt	VP
	2	2	1	hanna	NP
	2	2	2	geht	VP
	2	3	1	stimmt	VP
	2	3	2	das	NP
	3	1	1	läuft	VP
	3	1	2	das kind	NP

Da sich die vier Relationen in 1NF und 2NF befinden, bleibt noch zu prüfen, ob sie auch dem 3NF Kriterium genügen. Auch hier wollen wir uns kurz fassen, indem wir nicht für jede aufzeigen, daß es so ist, sondern uns auf TALK beschränken, die das 3NF Kriterium nicht erfüllt. TALK befindet sich nicht in 3NF, weil zwischen den Nichtschlüsselattributen WORTLAUT und WORTANZAHL funktionale Abhängigkeit besteht: Jedem Wert aus WORTLAUT ist genau ein Wert aus WORTANZAHL zugeordnet, d.h. es gibt keine Umstände, unter denen beispielweise der Wortlaut "ich ging" für WORTANZAHL mal den Wert "2" und mal den Wert "1" annehmen kann.

Erinnern wir uns, daß die Normalisierung ein Zerlegungsprozeß ist. Man würde jetzt so verfahren, daß man eine weitere Relation anlegt, etwa COUNT(WORTLAUT, WORTANZAHL), in der das Attribut WORTLAUT den Primärschlüssel bildet und in TALK den Fremdschlüssel. So zu verfahren, hätte zur Folge, daß TALK und COUNT zwei Relationen von annähernd gleicher Kardinalität wären, und die ist auf längere Sicht nicht gerade niedrig. Ob uns diese Zerlegung zum Vorteil gereicht, vermögen wir derzeit nicht zu beurteilen. Wir können allerdings beurteilen, zu welchen Speicheranomalien es aufgrund dieser Abhängigkeit kommen kann: So kann zum Beispiel das Tupel

(3, 1, I, läuft das Kind, 100, INT)

nicht in TALK eingeben werden, weil der Schlüsselwert "3,1" bereits existiert, wohl aber das Tupel

(7, 20, I, läuft das Kind, 100, INT)

wenn der Schlüssel "7,20" noch nicht existiert, und damit bestünde die Äußerung "läuft das kind" in einem Fall aus 3 Wortformen, im anderen aus 100. Wissend um dieses Problem wurde dennoch entschieden, TALK in 2NF zu belassen, weil die Anzahl der Wortformen pro Äußerung als marginal für die Projektfragestellung erachtet wird und weil die Verknüpfung der Relationen COUNT und TALK über die Werte des Attributs WORTLAUT extrem fehleranfällig ist.

Wir haben bei der Darstellung der Normalisierung nicht jedes Attribut explizit behandelt, sondern nur einige exemplarisch herausgegriffen. Daher kann u.U. der Eindruck entstehen, daß nicht alle Relationen in 2NF sind, oder daß z.B. PHRASEN nicht in 3NF ist, usw. Hier sei festgehalten, daß die Bestimmung der Beziehungen zwischen den Attributen grundlegend dadurch determiniert ist, wie jedes Attribut selbst be-

stimmt ist. So könnte man beispielsweise geneigt sein zu behaupten, daß zwischen dem Wortlaut einer Phrase und syntaktischer Kategorie funktionale Abhängigkeit besteht, wie es den linguistisch weniger Bewanderten unter uns immer wieder passiert ist. Und weil auch wir inzwischen verstanden haben, daß dem nicht so ist, hier ein Beispiel: "die Wüste Gobi" vs. "die wüste Gabi". Daß man den Wortlaut {wüste} mit zwei verschiedenen syntaktischen Kategorien klassifiziert liegt auf der Hand. So wie die Attribute bestimmt wurden, treffen unsere Feststellungen darüber, welche NF-Kriterien die Relationen erfüllen zu.

5.3.4. Die Relationen des Projekts und ORACLE

Wir haben die Projektdaten in Hinblick auf die drei Merkmale von Daten im relationalen Modell untersucht und versucht, den Ansprüchen des Modells so weit wie möglich gerecht zu werden. Ob wir die angestrebte Stukturierung in ORACLE umsetzen können, hängt vor allen Dingen davon ab, in welchem Ausmaß das DBMS ORACLE dem relationalen Modell gerecht wird, oder um es mit Rolland (1990, S. 11ff.) zu formulieren, "How relational is ORACLE?".

Auf jeden Fall wird die Datenstruktur unterstützt, d.h. die Forderung "All attribute values are *atomic*." (vgl. S 28). Desgleichen werden Datenmanipulationen unterstützt, die ihre Grundlage in der Mengenalgebra haben. Hierzu gehören SELECT, PROJECT, UNION, MINUS und TIMES, aus denen sich weitere Operationen ableiten lassen wie JOIN, INTERSECT, DEVIDE, etc. Damit ist ORACLE das, was als "relationally complete" bezeichnet wird, aber nicht "fully relational". "To be fully relational, a product must be relationally complete, allow domain definition *and* provide *entity integrity* and *referential integrity*." (Rolland, 1990, S. 16). ORACLE stellt zwar einige bereits definierte Domänen zu Verfügung, so z.B. *number*, *character*, *date*, etc., bietet dem Anwender aber nicht die Möglichkeit, selbst Domänen zu definieren. Die Datendefinitions- und -manipulationssprache, die ORACLE verwendet, ist SQL (Structured Query Language), und "SQL has no direct mechanism for defining or recognizing a key." (Rolland, 1990, S.17). Damit kann man Tabellen (hier ist dann doch wohl eher von Tabellen im Unterschied zu wohldefinierten Relationen zu sprechen) in die Datenbank einbringen, die ein und dasselbe Tuple mehrfach enthalten. Außerdem kann man von Tupeln einer Tabelle auf solche einer anderen referieren, die es nicht gibt [sic!]. Man ist also gut beraten, frühzeitig zu überlegen, wie man mit diesen Problemen umgeht, und ORACLE stellt hier wenigstens zwei Möglichkeiten zu Verfügung: die Verwendung von UNIQUE INDEXES für bestimmte Attribute oder die Definition der Schlüssel im Rahmen von SQL*Forms, einer Software, die die Basissoftware, das RDBMS, ergänzt.

Für die Projektanwendung haben wir entschieden, solange es eben geht, die Möglichkeiten von SQL*Plus und SQL*Forms zu nutzen. Wir haben also zunächst auf SQL*Plus-Ebene die Relationen eingerichtet, d.h. Relationen- und Attributnamen vergeben, die Attribute definiert und die Feldlänge festgelegt. Im Anschluß daran wurden über die Attributen DNR, TNR, ANR und PNR bzw. über Kombinationen aus diesen UNIQUE INDEXES gebildet, so daß sie als Primärschlüssel fungieren können.

Ein Attribut kann als NUMBER oder CHARACTER definiert werden, d.h. die Werte können numerischer oder alphanumerischer Art sein. Alle Attribute der Relationen, deren Wertebereich ausschließlich numerische Werte umfaßt {1, 2,..., n} wurden als NUMBER definiert, die übrigen als CHARACTER. Für jedes Attribut kann NULL festgelegt werden, d.h. "missing information" ist zulässig. Definiert man ein Attribut als NOT-NULL, dann ist "missing information" nicht zulässig. Sämtliche Schlüsselfelder wurden als NOT-NULL definiert, also auch DNR in der Relation TEXTE. Als NOT-NULL wurden außerdem alle Attribute der Relationen DESIGN und TEXTE definiert, sowie SPRECHER und WORTLAUT in der Relation TALK und WORTLAUT_PHR in PHRASEN. Für die Attribute, die als Primärschlüssel fungieren sollen, wurden UNIQUE INDEXES kreiert.

6. Arbeiten mit den Daten in ORACLE

In Abschnitt 4.2 wurden die Anforderungen an das Datenbanksystem ORACLE aufgezeigt. Abschnitt 5 legte die Struktur der Daten in der Datenbank bzw. den einzelnen Tabellen dar. In diesem Abschnitt wollen wir nun darauf eingehen, welche Komponenten von ORACLE wir eingesetzt haben, um die Daten des Projekts zu verarbeiten. Zunächst formulieren wir, welche Anforderungen wir an eine Datenbank stellen. Insbesondere ist es das Ziel von Abschnitt 6.1, die einzelnen Arten von Anwendungen aufzuzeigen. In Abschnitt 6.2 wird kurz die ORACLE-Komponente SQL*Plus vorgestellt. In Abschnitt 6.3 beschreiben wir die Ausgabe der Daten in Form von gedruckten Listen. Abschnitt 6.4 zeigt, wie wir "nicht ORACLE gemäß"-EDV-erfaßte Daten in die Datenbank eingespielt haben. Der Abschnitt 6.5 ist eine kurze Einführung in die ORACLE-Komponente SQL*Forms und 6.6 schildert an einigen Beispielen Probleme, die sich uns beim Arbeiten mit SQL*Forms stellten, und unsere Lösungen dazu.

6.1. Arten von Anwendungen

a) Recherchen

Sinn und Zweck der Datenbank ist es, einen möglichst schnellen Zugriff auf die Daten zu ermöglichen. Der Zugriff kann sich auf die Gesamtheit der Daten beziehen, oder als Recherche mit bestimmten Auswahlkriterien erfolgen. Wir differenzieren 5 Arten von Suchen in der Gesamtheit der Daten:

(a) Suchen in den Ordnungsebenen, Diskurs, Texte, Äußerung und Phrasen, d.h. in den Tabellen DESIGN, TEXTE, TALK und PHRASEN.
(b) Suchen innerhalb einer oder mehrerer Ordnungsebenen nach bestimmten Kategorien oder Kombinationen von Kategorien.
(c) Suchen innerhalb einer oder mehrerer Ordnungsebenen nach Variationen einer oder mehrerer Kategorien unter Konstanthaltung der übrigen.

Hinzu kommen zwei weitere Arten von Recherchen, die nur bei der Erfassung der Daten benötigt werden:

(d) Abfragen, die es einem Bearbeiter ermöglichen, seine Eingaben zu überprüfen und gegebenenfalls zu korrigieren.
(e) Abfragen zum Zweck der Ergänzung vorhandener Daten.

Die Ergebnisse von Recherchen sollen auf dem Bildschirm ausgegeben oder auf einem Drucker gedruckt werden.

b) Eingabe

In der Vergangenheit wurden die mit dem Tonband erfaßten Gespräche als Transkripte erfaßt und auf Papier ausgedruckt. Die Einteilung in Äußerungseinheiten fand auf dem Papier statt, und die zugehörigen Kategorien wurden entweder auf dem Ausdruck oder auf Karteikärtchen festgehalten. Die Eingabe der Projektdaten soll möglichst einfach zu handhaben sein. Um die Umsetzung von der "Papierarbeit" auf die EDV-gestützte Erfassung so transparent wie möglich zu gestalten, soll die Eingabe ähnlich dem Ausfüllen eines Formblattes ablaufen.

Zunächst war die Planung, zwei Versionen von Formblättern zu erstellen, getrennt nach der Kompetenz des Bearbeiters:

(a) Erfassung von Rohdaten. Dies umfaßt im wesentlichen das Feld Wortlaut in den Tabellen TALK und PHRASEN.
(b) Erfassung der qualitativen Analyseergebnisse als Ergänzung zu (a), also alle anderen Kategorien der Ordnungsebenen Äußerung und Phrasen.

Version (b) könnte auch noch weiter untergliedert werden. Z.B. könnte man zunächst ein Formblatt entwickeln, in dem, zusätzlich zu den schon in (a) eingegebenen Daten, noch genau eine weitere Kategorie erfaßt werden kann, z.B. Diskursfunktion. Im nächsten Schritt ein Formblatt, in dem zu (a) und der Diskursfunktion noch Illokution eingegeben werden kann. usw. Dies wäre nützlich, falls nicht alle Klassifikationen in einem Analyseschritt vorgenommen werden sollten. Beim späteren Arbeiten mit den Formblättern stellte sich jedoch heraus, daß es zum einen auch für den Eingebenden von (a) sinnvoll ist, wenn er alle Klassifikationen vor sich sieht (das Überspringen von nicht auszufüllenden Feldern ist problemlos), zum anderen reduziert dies die Zeit, die jedesmal wieder aufgewendet werden muß, um eine abgeänderte Version eines Formblattes zu erstellen und auszutesten.

6.2. SQL*Plus

SQL*Plus dient zur Kommunikation zwischen dem Benutzer und dem Datenbankkern, dem RDBMS (Relational Database Management System). Man bedient sich dazu einer Spache SQL - Structured Query Language, die eine Reihe von Befehlen zum Datenzugriff und zur Datenmanipulation zur Verfügung stellt. Die Befehle unterliegen strengen Konventionen, die ein Benutzer genau kennen muß, um seine Absicht mit der Datenbank realisieren zu können.

Größere Dateneingaben und Korrekturen sind mittels der standardisierten SQL-Kommandos, die über SQL*Plus eingegeben werden können, umständlich und fehleranfällig. Es wird die genaue Kenntnis der Spaltennamen vorausgesetzt, und bei einigen Kommandos ist die Reihenfolge der Spaltenangaben wesentlich. Ferner müssen alle einzugebenden Werte, die aus alphanumerischen Zeichen bestehen, in Hochkommata eingeschlossen werden. Ein optisches Hilfsmittel, um eine Übersicht über die Spalten zu bekommen, gibt es nicht.

Wir haben im wesentlichen mit SQL*Plus nur die Tabellen DESIGN, TEXTE, TALK und PHRASEN definiert und den einzelnen Spalten die notwendigen Attribute "CHARACTER" vs. "NUMBER", "NULL" bzw. "NOT NULL" zugewiesen, die Anzahl der Zeichen pro Spalte vergeben und UNIQUE INDEXES festgelegt (vgl. Abschnitt 5.3.4).

Um das Datenbank-Management, also z.B. regelmäßige Sicherungskopien anzufertigen, Plattenplatz für die Datenbank zur Verfügung zu stellen etc., mußten wir uns nicht kümmern. Dies übernimmt auf der Großrechenanlage das Rechenzentrum.[24)]

6.3. Drucken der Ergebnisse

Um die Ergebnisse einer Recherche auszudrucken, bietet ORACLE als weitere Komponente SQL*Report Writer an. Man kann einzelne Spalten oder Zeilen aus einer Tabelle auswählen und die Ausgabe formatieren. SQL*Report Writer benutzt eine eigene Sprache, in der man die Formatierungsangaben mitteilt. Wir haben uns bisher mit den Möglichkeiten, die SQL*Plus zum Drucken und Formatieren bereitstellt, zufriedengegeben. Dies sind im wesentlichen die Festlegung einer Spaltenbreite, sowie Zeilenumbrüche und Zeilenvorschübe. So genügt es z.B. für einen Kontrollabdruck qualitativer Analyseergebnisse, nur die Satzanfänge zu sehen, also das Feld Wortlaut auf 40 Zeichen zu begrenzen.

---

24) An dieser Stelle möchten wir Rolf Bogus vom Rechenzentrum der Universität Heidelberg für seine Unterstützung und Beratung danken.

6.4. Einspielen von ASCII-Daten

Da natürlich Textdaten schon im Projekt vorhanden waren, bevor es die Datenbank gab, und zu anderen Zwecken Texte in WORD oder WordPerfect erfaßt werden, ist es sinnvoll, diese in Äußerungseinheiten gegliederten Texte in die Datenbank in das Feld WORTLAUT der Tabelle TALK zu übernehmen. Dies geschieht folgendermaßen: Der Text wird von WORD aus als ASCII-Text abgespeichert, per Filetransfer auf den Großrechner übertragen und hier mit Hilfe von SQL*Load eingespielt. SQL*Load nimmt ferner folgende Eingaben automatisch vor: Die Identifizierungsnummern DNR und TNR sowie der Informantenname INAME werden einmal in die Tabelle TEXTE eingetragen. Die Identifikationsnummer TNR für die Tabelle TALK wird als Konstante angegeben, da sie über den ganzen Text fest ist. In der Tabelle TALK wird neben TNR auch ANR eingetragen, welche mit Hilfe eines Zählers automatisch erzeugt wird. Das Feld SPRECHER der Tabelle TALK wird, falls kein Wert in den Eingabedaten steht, auf einen Dummy-Wert gesetzt, der später mit SQL*Forms korrigiert werden muß. Dies ist notwendig, da das Feld SPRECHER als NOT NULL definiert ist.

Beim Einspielen von ASCII-Daten finden nur Integritätsprüfungen auf Tabellenebene statt, d.h. es werden nur die Attribute geprüft, die beim Anlegen der Tabellen in SQL*Plus für die Spalten definiert wurden (CHARACTER vs. NUMBER, NULL bzw. NOT NULL und UNIQUE INDEX). Groß-/Kleinschreibung z.B. des Informantennamens ist, wenn die Datenbank korrekt bleiben soll, vorher in einem Editor genau zu prüfen.

6.5. SQL*Forms

SQL*Forms ist ein Maskengenerator, der es gestattet, menügesteuert Masken für vorhandene Tabellen zu entwickeln. Diese Masken werden Formblätter oder kurz Forms genannt. Die Struktur der Datenbanktabellen wird auf übersichtliche Art und Weise analog einem Formblatt präsentiert. Es ermöglicht Eingabe und Veränderung von Daten ohne die Kenntniss der zugrundeliegenden SQL-Befehle. Ebenso ist das Abfragen von Daten mit SQL*Forms wesentlich erleichtert.[25] Die Spalten der Tabellen erscheinen als Felder, die in Länge und Typ nur bestimmte Eingaben zulassen. Der Grund, warum solche Masken nicht mit einer herkömmlichen Programmiersprache wie C oder Pascal realisiert werden, liegt in der Portabilität der in SQL*Forms entwickelten Masken. So ist es möglich, die auf dem Großrechner entwickelten Masken später identisch auf dem PC einzusetzen.

SQL*Forms bietet vielfache Möglichkeiten zur Formblattgestaltung. Unabhängig von den mit SQL*Plus definierten Attributen für eine Datenbankspalte werden in SQL*Forms eine Reihe weiterer Attribute zur

---

25) So ganz ohne SQL-Kenntnisse kommt man hierbei jedoch auch nicht aus, denn z.B. ODER-Verknüpfungen sind nur über eine Kommandozeile, in die man das gewünschte SQL-Statement hineinschreibt, möglich.

Verfügung gestellt, z.B. UPPERCASE oder PRIMARY KEY. Man kann zu jedem Feld eine Online-Hilfe generieren. Des weiteren besteht die Möglichkeit, Trigger - einzelne Kommandos oder Kommandofolgen, die zu einem bestimmten Zeitpunkt ausgeführt werden - zu verwenden. Darüberhinaus sieht SQL*Forms noch den Einsatz von User Exits vor, d.h. von Programmen in herkömmlichen Programmiersprachen.

Zunächst erscheint SQL*Forms recht einfach, es ist aber doch ziemlich gewöhnungsbedürftig, so muß man einige Menüs durchlaufen, um die Definitionen für die Felder der Maske vornehmen zu können. Als Beispiel für die Umständlichkeit von SQL*Forms kann man folgendes anführen: um ein abgeändertes Feldattribut in einer Form zu verankern, muß man nach dem Abändern fünfmal die Taste für "Beenden mit Sichern" drücken, um dann die Maske neu generieren zu können.[26] Eine Alternative zum Abändern in SQL*Forms besteht darin, diese Änderungen in dem automatisch von ORACLE erzeugten File vorzunehmen, in dem die Maske dokumentiert ist. In der uns vorliegenden Version von SQL*Forms führten jedoch schon relativ geringe Änderungen dazu, daß die Form nicht mehr lauffähig war, so daß wir diesen Weg nicht weiter verfolgt haben. Auch im Hinblick auf das Triggerkonzept verführt SQL*Forms mit seiner Einfachheit zu schwer lesbarer, unstrukturierter Programmierung. Durch die Menüsteuerung verliert man leicht den Überblick, wo welcher Trigger eingebaut wurde. In vielen Büchern wird davor gewarnt, sofort mit dem Erstellen von Masken loszulegen. Trotz aller Planung kommt man jedoch in der Realität an einigen Punkten nicht darum herum, das Ergebnis wegzuwerfen und neu zu beginnen. Auch im Projekt stellte sich an mehreren Stellen heraus, daß es zeitsparender ist, die Maske komplett neu zu generieren, anstatt die vorhandene weiter abzuändern. Es ist dann nur noch Fleißarbeit, sich durch die einzelnen Menüs "durchzukämpfen".

SQL*Forms Anwendungen unterliegen einer hierarchischen Struktur: Die oberste Stufe dieser Struktur bildet die Form, sie ist in sich logisch abgeschlossen. Eine Form setzt sich aus einem oder mehreren Blöcken zusammen. Die Daten eines Blockes werden in Feldern dargestellt, d.h. die Felder stellen die unterste Stufe der Hierarchie dar. Die übrigen Gestaltungsmerkmale, wie Feldbezeichnungen, Rahmen, etc. sind keine Bestandteile dieser Hiearchie.

6.6. Einzelne Probleme in SQL*Forms

Ohne jetzt auf die einzelnen von uns entwickelten Forms einzugehen, möchten wir an dieser Stelle ein paar von den Problemen aufzeigen, die sich uns beim Arbeiten mit SQL*Forms stellten. Die Probleme sind zum einem in den Hardware-Voraussetzungen begründet, die wir hier in Heidelberg hatten und haben, denn obwohl das Datenbanksystem ORACLE auf sehr vielen verschiedenen Maschinen mit verschiedenen Betriebsystemen von der Kommandosprache her identisch abläuft, gibt es doch Unterschiede in der Benutzung von ORACLE (vgl. dazu Problem LONG-Felder). Zum anderen sind Probleme in ORACLE selbst vorge-

---

26) Dies soll in der nächsten Version von SQL*Forms (Version 3) besser sein.

geben, da das im englischen Sprachraum entwickelte Produkt die Besonderheiten der deutschen Sprache nicht berücksichtigt.

a) LONG-Felder

ORACLE bietet für Textdaten Felder mit einer maximalen Länge von 240 Zeichen an. Es gibt zwar auch noch Felder für (von ORACLE sogenannte) Rohdaten, der Länge 64 kByte, in diesen Feldern kann man aber keine Suchoperationen durchführen. Zudem können sie nicht in SQL*Forms eingebunden werden. Für unsere Äußerungseinheiten besteht also eine obere Grenze von maximal 240 Zeichen. Das klingt zunächst recht viel, denkt man aber daran, daß im Deutschen zusammengesetzte Hauptwörter recht schnell 15-20 Zeichen lang sind (als Beispiel diene das Wort Äußerungseinheiten mit 18 Zeichen) und ferner noch Transkriptionszeichen in die Äußerungen eingefügt werden, so stellt diese Obergrenze doch eine erhebliche Einschränkung dar. Wir haben dieses Problem umgangen, indem wir die Transkriptionszeichen in den Äußerungen weglassen.[27)] Die wenigen Äußerungen, die trotzdem noch länger als 240 Zeichen sind, werden beim 239sten Zeichen abgeschnitten und an der 240sten Stelle markiert ein Sonderzeichen, daß die Äußerung hier eigentlich weitergeht.

Nachdem wir den Kompromiß, uns auf 240 Zeichen für das Feld WORTLAUT zu beschränken, geschlossen hatten, stellten wir fest, daß das Arbeiten mit Feldern dieser Länge in SQL*Forms nicht optimal gelöst ist. In einer SQL*Forms-Maske ist es nicht möglich, ein Feld umzubrechen. D.h. ein Feld, welches länger ist als die Bildschirmbreite, ragt über den rechten Bildschirmrand hinaus. Unsere Bildschirme haben eine Breite von 80 Zeichen, davon gehen 8 Zeichen für die Feldbezeichnung WORTLAUT ab, ein Leerzeichen zur Trennung von Feldbezeichnung und Feld, und übrig bleiben 71 Zeichen, die man von dem 240-Zeichen langen Feld auf dem Bildschirm sieht. Wir können ein Feld, welches länger ist als die gegebene Bildschirmbreite, also nie in seiner Gesamtheit auf dem Bildschirm sehen.

Um eine Verschiebung des Bildschirmfensters bei der Eingabe zu erreichen, muß man auf dem Großrechner eine Funktionstaste drücken. Man sieht dann den neuen Bildschirmausschnitt und ein paar Zeichen vom vorherigen Ausschnitt.

Wir haben mehrere Alternativen zu diesem Problem durchdacht. Eine diskutierte Lösung war, in den Forms vorzugaukeln, daß man nicht ein Feld der Länge 240 Zeichen hat, sondern z.B. vier Teilfelder der Länge 60. Über entsprechende Programmierung würden diese Einzelfelder vor dem Abspeichern in die Datenbank miteinander zu einem Gesamtfeld verkettet, so daß ein 240 Zeichen langes Feld entsteht. Dies führt zu unlösbaren Folgeproblemen, denn ein Löschen von Textteilen mit automatischem Nachrücken der Zeichen von rechts ist hierbei nicht möglich. Man könnte vorschreiben, daß immer nur ganze Worte in ein Teilfeld

---

28) Das Weglassen der Transkriptionszeichen hat des weiteren einen großen Vorteil bei Suchen im Wortlaut einer Äußerung: Wir müssen keine Rücksicht darauf nehmen, ob Abweichungen von der gesuchten Zeichensequenz durch die Transkiptionszeichen entstehen.

geschrieben werden dürfen und so die Felder nicht ganz füllen. Einfügen ist überhaupt nur möglich, wenn entweder die Einfügung so gering ist, daß sie gerade in das Teilfeld hineinpaßt, oder alle nach der Einfügung folgenden Zeichen werden neu eingegeben. Dies ist ein fehlerträchtiges, mühseliges Verfahren.

Um die Eingabe für die langen Felder zu erleichtern, haben wir probiert, aus der Form heraus, beim Erreichen eines langen Feldes einen externen Editor aufzurufen, in dem der Bearbeiter dann über mehrere Zeilen hinweg seinen Text eingibt und der abgespeicherte Editor-File sofort in die Datenbank importiert wird. Dieser Weg erwies sich in der Realisierung auf dem Großrechner als ziemlich schwierig (auch zum Teil in der gegebenen Version von SQL*Forms begründet), so daß wir auch diese Idee nicht weiter verfolgt haben.

Auch andere denkbare Lösungen zogen entweder massive Folgeprobleme nach sich, oder waren mit den Hardwarevoraussetzungen und unserem Wissen nicht lösbar, so daß wir uns hier mit den gegebenen Möglichkeiten abgefunden haben.

b) Umlaute

Das ewige Problem der aus dem englischsprechenden Raum importierten elektronischen Datenverarbeitungsgeräte sind die Umlaute. Auf einem PC mit deutscher Tastatur sofort verwendbar, sind sie auf einem Großrechner mit amerikanischer Tastatur nicht automatisch realisiert. Da ORACLE ein in sich abgeschlossenes System ist, welches die Bildschirm- und Plattenverwaltung selbst übernimmt, nutzen alle Tricks, mit denen man sonst Umlaute auf dem Großrechner eingeben und darstellen kann, überhaupt nichts. In ORACLE selbst ist das Eingeben von Sonderzeichen möglich, wenn sie direkt über die Tastatur eingegeben werden können.

Ein theoretisch gangbarer Weg wäre eine Umsetzung von definierten Codes bei den eingegeben Daten vor dem Schreiben in die Datenbank (also vor dem COMMIT). Z.B. könnte man definieren "a wird in ä umgesetzt etc.". Eine Rückumsetzung müßte vor jedem Suchbefehl stattfinden. Diese Umsetzung, zunächst einmal auf SQL*Forms Ebene realisiert, würde aber z.B. bei SQL*Plus nicht wirken. Da ferner SQL*Forms an sich auf dem Großrechner schon nicht besonders schnell ist, haben wir auf diese massive Bremse verzichtet, zumal die erfaßten Daten gemischtsprachlich (englisch, deutsch) sind.

c) Felddarstellung

Eine weiteres Problem auf dem Großrechner ist die Felddarstellung in SQL*Forms. Erst unter Angabe von Parametern ist es möglich, die Felder unterstrichen darzustellen, so daß man die Feldbegrenzung sieht.[28] Eine Helldarstellung der Felder, die sich auch noch invertiert, wenn der Cursor in das Feld springt, ist mit

---

28) Dies ist in der nächsten Version von SQL*Forms (Version 2.3) besser.

der jetztigen SQL*Forms Version nicht möglich. Eine optische Hilfe bei der Bearbeitung der Masken durch die Felddarstellung kann also nicht gegeben werden.

7. Die Forms

Die Entwicklung der Forms, die wir hier darstellen werden, war grundlegend von der Überlegung geleitet, daß alle Daten über den Bildschirm einzugeben seien. Im Verlauf der Entwicklung hat sich gezeigt, daß bestimmte Eingaben recht fehleranfällig sind. Hierzu gehören vor allen Dingen die Werte von Klassifikationskategorien, bei denen mnemotechnische Kriterien wichtiger schienen als orthographische, sowie die numerischen Werte der Schlüsselfelder. Das erste Problem ist ohne weiteres lösbar, indem man einen eingegebenen Wert gegen die Liste der erlaubten Werte prüfen läßt, d.h. eine Eingabe wie z.B. "MANIPROP" anstatt "PROP+MANIP" wird automatisch als nicht-zulässiger Wert reklamiert, was natürlich für echte Fehlklassifikationen nicht machbar ist. Das zweite Problem hängt wesentlich mit den Darstellungsmöglichkeiten der Forms zusammen, ist aber im Ansatz, wenn auch noch nicht zu unserer vollen Zufriedenheit, gelöst.

Für die Relationen haben wir je eine Form entwickelt, über die Eingaben, Änderungen und rudimentäre Abfragen vorgenommen werden können. Wählt man für die Entwicklung der Forms das von SQL*Forms angebotene Standardverfahren (default), dann enthält die erste Version einer Form als Felder die Attribute der Relation, für die sie erstellt wird. Die Felder sind bereits mit den Spezifikationen versehen, die für die Attribute beim Einrichten der Relation vorgenommen wurden. So ist z.B. ein Attribut, das über SQL*Plus als NUMBER und NOT NULL definiert wurde, beim entsprechenden Feld in der default-Form als NUMBER und MANDATORY spezifiziert. Diese Spezifikationen können geändert, aufgehoben oder ergänzt werden. Solche Modifikationen sind aber immer nur wirksam, wenn die Datenbearbeitung mit eben dieser Form erfolgt. Wir werden nachfolgend darstellen, was wir von den default-Forms übernommen und was wir geändert haben. Dem können wir vorausschicken, daß wir für alle Forms die Validierung, die bei der default-Entwicklung auf Blockebene erfolgt, auf Feldebene geändert haben, zum einen, um *fieldtriggers* einsetzen zu können, zum anderen kann dadurch ein Feld erst verlassen werden, wenn es korrekt bearbeitet wurde.

7.1. d_entry und DESIGN

Die Bildschirmausgabe der Form **d_entry** für DESIGN ist in Abbildung 4 wiedergegeben. Sie besteht aus dem Block DESIGN.

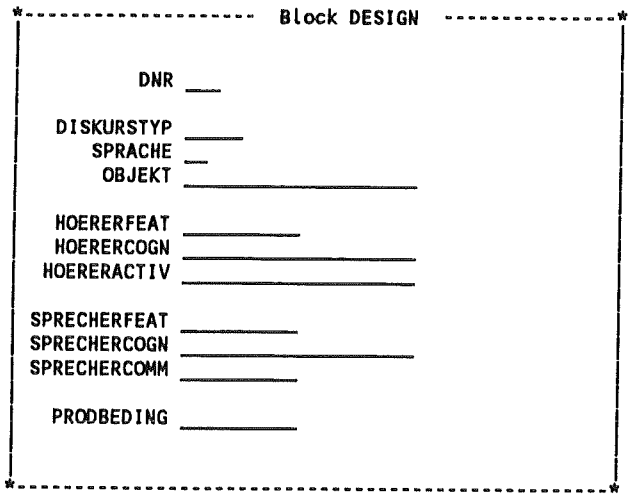

Abbildung 4: *Bildschirmausgabe der Form d_entry*

a) Beschreibung der Felder

Die Form umfaßt eine Bildschirmseite und erlaubt nur eine Zeile (*record*) darzustellen (vgl. die Form für TEXTE, S.48). Das Feld DNR ist auf Formebene bereits als NUMBER, die übrigen Felder sind als CHAR definiert. Zudem ist jedes Feld als DATABASE FIELD, DISPLAYED, INPUT ALLOWED, QUERY ALLOWED, UPDATE ALLOWED und MANATORY spezifiziert. Zunächst wurde für alle Felder UPDATE ALLOWED aufgehoben, um zu verhindern, daß DESIGN-Information versehentlich geändert wird, da dies negative Folgen für große Datenbankausschnitte hat.[29] Für DNR wurde NUMBER in RINT geändert, d.h. in DNR dürfen nur ganzzahlige numerische Werte vorkommen; sie werden rechtsbündig geschrieben. Für die CHAR-Felder wurde als weitere Spezifikation UPPERCASE hinzugenommen. AUTOSKIP wurde dort gewählt, wo die Werte eines Attributs gleich lang sind, z.B. bei DISKURSTYP. Die Felder sind von DNR bis PRODBEDING fortlaufend numeriert (SEQ# 1 bis 11).

b) Funktionsweise

Nach dem Aufruf der Form befindet sich der Cursor im Feld DNR. Die Eingabe beginnt bei diesem Feld, und der Wechsel zum nächsten Feld ist erst möglich, wenn eine Zahl eingegeben wurde. Die nächste Eingabe muß im Feld DISKURSTYP vorgenommen werden. Sie ist im Prozedere für dieses und die nachfolgenden Felder gleich. Sie kann frei erfolgen, d.h. der jeweilige Wert wird in das Feld eingetragen. Der Eintrag kann in Groß- oder Kleinschreibung erfolgen. Letztere wird beim Wechsel zum nächsten Feld in Großschreibung umgesetzt. Der Wechsel zum nächsten Feld löst zudem einen Trigger aus, der prüft, ob der eingegeben Wert formal zulässig ist. Um diese Prüfung zu ermöglichen, wurde eine Tabelle HILFE_DT(DISKURSTYP) angelegt, die als Werte BESCH, ERZAE und INSTR enthält. Der Versuch, etwa BESCC einzugeben, wird mit der Fehlermeldung "Den Wert gibt es nicht, bitte korrigieren" quittiert. Zugleich wird der Wechsel zum nächsten Feld unterbunden und zwar solange, bis der eingegebene Wert

29) Für gegebenenfalls notwendig werdende updates steht die gleiche Form zur Verfügung, mit UPDATE ALLOWED bei allen Feldern. Für diese Form ist die Zugriffsberechtigung eingeschränkt.

korrekt ist.[30] Alternativ zur freien Eingabe kann man sich die Werte aus HILFE_DT anzeigen lassen und den jeweils passenden übernehmen.

Sind in der Form alle Felder ausgefüllt, kann entweder NEXT RECORD für eine weitere Eingabe angefordert werden, oder man gibt die Einträge mit COMMIT TRANSACTION an DESIGN ab. Sollte der bei DNR eingegebene Wert bereits in der Relation vorkommen, wird die Eingabe zurückgewiesen und eine ORACLE-Fehlermeldung auf dem Bildschirm ausgegeben, die man sich spezifizieren lassen kann; ohne Kenntnis der Primärschlüsselfunktion wird aus der Fehlermeldung nicht ersichtlich, daß der Fehler mit DNR zusammenhängt. Dies gilt auch für die entsprechende Fehlermeldung, die ausgegeben wird, wenn man in SQL*Forms PRIMARY KEY und CHECK FOR UNIQUE INDEX spezifiziert, weshalb wir darauf verzichtet und die ORACLE-Fehlermeldung im Eingabemanual erläutert haben.

c) Triggers und Hilfen

Die Eingabe pro Feld wird über den Feldtrigger POST-CHANGE, der aus einem Triggerschritt besteht, geprüft:

```
SELECT 'X'
FROM HILFE_DT
WHERE HILFE_DT.DISKURSTYP = :DESIGN.DISKURSTYP
```

Damit wird geprüft, ob die Tabelle HILFE_DT einen Wert enthält, der mit dem Wert im Feld DISKURSTYP des Block DESIGN übereinstimmt. Wenn das nicht der Fall ist, wird eine Fehlermeldung auf dem Bildschirm ausgegeben, die man im *trigger step window* selbst formulieren kann.

Für jedes Feld kann die Eingabe durch online-Hilfen unterstützt werden. Hierzu wird von SQL*Forms ein *window* SPECIFY VALIDATION zur Verfügung gestellt. Von den dort gebotenen Möglichkeiten haben wir HELP und LIST OF VALUES genutzt. Bei HELP kann ein Text eingegeben werden, der entweder automatisch gezeigt wird, wenn AUTOMATIC HELP für das Feld spezifiziert wurde, andernfalls kann die Information über die HELP-Taste abgerufen werden. Für den Text, der bei HELP eingegeben werden kann, steht lediglich eine Bildschirmzeile zur Verfügung. Wir haben hier "Werte aus dem Manual entnehmen oder anzeigen lassen mit PF15" als Text eingegeben. Der Tastenschlüssel PF15 steht für den Befehl LIST FIELD VALUES, womit wir beim zweiten Teil der Hilfen wären.

Um die Liste der erlaubten Werte zu spezifizieren, gibt man den Tabellennamen ein, um bei unserem Beispiel zu bleiben, HILFE_DT, und den Attributnamen DISKURSTYP. Man kann sich die Werte nun nacheinander anzeigen lassen und den passenden übernehmen. Mit Hilfe welcher Tasten die Übernahme erfolgt, ist dann wieder im Manual erläutert.

---

30) Falsche Eingaben, wie BESCH anstatt INSTR, sind keine formalen Fehler sondern inhaltliche, für die es keine Prüfmechanismen gibt und aus ersichtlichen Gründen auch in absehbarer Zeit nicht geben wird.

Was wir hier am Beispiel von DISKURSTYP beschrieben haben, gilt auch für die übrigen Attribute bzw. Felder, d.h. wir haben den Bestand an Relationen (DESIGN, TEXTE, TALK und PHRASEN) um die Tabellen HILFE_DT für DISKURSTYP, HILFE_S für SPRECHER bis hin zu HILFE_PB für PRODBEDING erweitert, die entsprechenden Triggers gesetzt und die Wertelisten bereitgestellt.

7.2. txt_entry und TEXTE

Die Form **txt_entry** für TEXTE (Abbildung 5) besteht aus dem Block TEXTE, umfaßt ebenfalls eine Bildschirmseite und erlaubt die Darstellung mehrerer *records*.

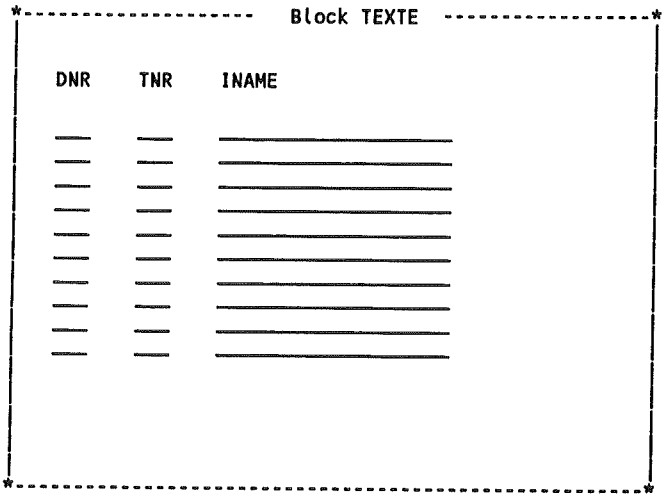

Abbildung 5: *Bildschirmausgabe der Form txt_entry*

a) Beschreibung der Felder

Hier gilt im wesentlichen das gleiche wie für die Felder der Form d_entry. DNR und TNR wurden von NUMBER zu RINT spezifiziert und INAME als UPPERCASE. UPDATE ALLOWED wurde nicht aufgehoben.

b) Funktionsweise

Nach dem Aufruf der Form befindet sich der Cursor in der ersten Zeile auf dem Feld unter DNR. Die Eingabe erfolgt bei diesem und den beiden anderen Feldern ausschließlich frei, d.h. es werden keine Wertelisten angeboten. Das Anfordern von online-Hilfe resultiert in der Aufforderung "Bitte Wert von der Vorlage übernehmen".

c) Triggers und Hilfen

Beim Feld DNR ist wieder ein POST-CHANGE-Trigger gesetzt, der prüft, ob der eingegebene Wert in DESIGN vorhanden ist. Er besteht aus dem Triggerschritt:

```
SELECT 'X'
FROM DESIGN
WHERE DESIGN.DNR = :TEXTE.DNR
```

Mit diesem Trigger wird - zumindest während der Arbeit mit der Form - der referentiellen Integrität insoweit Rechnung getragen, als es nicht möglich ist, Information in die Relation TEXTE einzugeben, für die die DESIGN-Information nicht vorhanden ist. Entsprechende Versuche werden mit der Meldung "Bitte die Eingabe abbrechen und DESIGN-Information des Textes eingeben." quittiert.

7.3. t_entry und TALK

Die Form **t_entry** (Abbildung 6) für TALK besteht aus den beiden Blöcken AUTOMAT und TALK, umfaßt eine Bildschirmseite und erlaubt im Block TALK lediglich ein *record* darzustellen. Diese wäre auch nicht anders, wenn wir auf den Block AUTOMAT verzichtet hätten. Die Tatsache, daß nur ein *record* im Block TALK vorkommen kann, hat uns dazu veranlaßt, den Block AUTOMAT aufzunehmen. Es verhält sich nämlich so, daß nach der Anforderung NEXT RECORD die Eingabefelder aller Attribute leer sind, d.h. man hat keine Information mehr bezüglich der zuvor durchgeführten Eingabe, und die Erfahrung im Umgang mit einer Form, die lediglich den Block TALK enthielt, hat gezeigt, daß bereits nach kurzer Zeit Probleme bei der Eingabe auftreten, wie etwa die Frage: "Habe ich die Äußerung schon eingegeben, welche Textnummer habe ich ihr zugeordnet, welche Äußerungsnummer?". Mit Hilfe des Blocks AUTOMAT wurde versucht, derartigen Problemen zu begegnen.

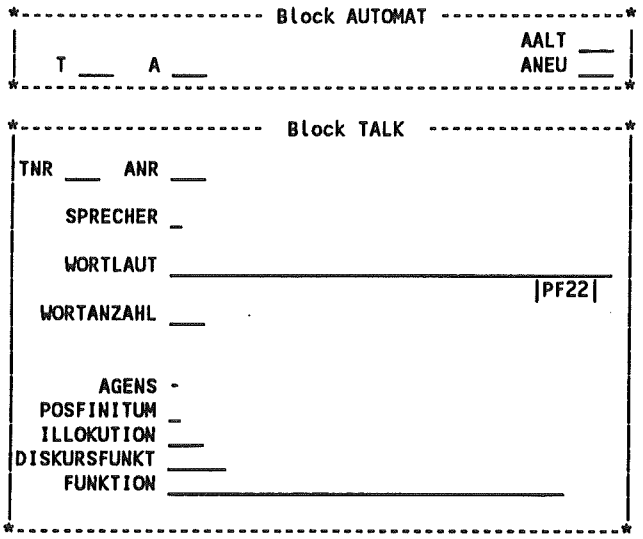

Abbildung 6: *Bildschirmausgabe der Form t_entry*

a) Beschreibung der Felder

Dem Block AUTOMAT (Seq# 2) liegt die Tabelle TAUTOMAT(R, T, A) zugrunde, in der die Attribute als "NUMBER" und "NOT NULL" definiert sind und R als UNIQUE INDEX. In TAUTOMAT wurde ein *record* (R =1, T = 0, A = 0) eingegeben. In SQL*Forms wurde NUMBER in INT geändert. Für R wurde lediglich DATABASE FIELD beibehalten. Für T und A wurde AUTOMATIC HELP spezifiziert.

Die beiden Felder AALT (Nummer der Äußerung, die zuletzt eingegeben wurde) und ANEU (Nummer der Äußerung, die als nächste einzugeben ist) gehören zu keiner Tabelle und dienen nur zur Darstellung bestimmter Informationen, die die Eingabe unterstützen, weshalb für sie lediglich DISPLAYED spezifiziert wurde.

Die Felder im Block TALK (Seq# = 1) haben von SPRECHER bis FUNKTION als Seq# "1" bis "8", TNR hat "9" und ANR hat "10". Alle Felder gehören zur Relation TALK. Dort sind TNR, ANR und [illegible]TAN-ZAHL als NUMBER definiert, die übrigen Felder als CHAR. TNR, ANR, SPRECHER und W[illegible]LAUT sind zusätzlich als "NOT NULL" festgelegt. Auf Form-Ebene sind die NUMBER-Felder als RINT definiert. Für alle CHAR-Felder außer WORTLAUT gilt zudem UPPERCASE. Für TNR und ANR wurden in der Form nur die Spezifikationen DATABASE FIELD und QUERY ALLOWED beibehalten.

b) Funktionsweise

Der Cursor befindet sich nach dem Aufruf der Form im Block TALK auf dem Feld SPRECHER. Im Block AUTOMAT wird beim Feld AALT ein Wert "n" angezeigt, im Feld ANEU der Wert "n+1". Das Kommando NEXT BLOCK setzt den Cursor auf das Feld T im Block AUTOMAT. Mit EXECUTE QUERY geht man in den Abfragemodus und erhält zunächst die Information "Do you want to commit the changes you have made? Y".[31] Hier gibt man N ein. Danach werden in die Felder T und A aus der Tabelle TAUTOMAT die dort vorhandenen Werte <t,a> eingelesen. Diese sind zu Beginn der Eingabe <0,0> (wenn die Relation TALK noch keine Einträge enthält). Im Feld AALT wird dann "0" angezeigt und "1" im Feld ANEU. Jetzt ist ein UPDATE durchzuführen und zwar zu Beginn der Eingabe nur für das Feld T. Man gibt hier die Nummer des Textes (z.B. "1") ein, für den im Block TALK Einträge vorgenommen werden sollen.

Beim Feld T wurde wieder der bereits bei der Form d_entry erörterte POST-CHANGE Trigger eingesetzt, über den geprüft wird, ob der für T eingegeben Wert "korrekt" ist, d.h. ob die Relation TEXTE den hier vergebenen Wert bereits als Textnummer enthält. Wenn ja, dann wird diese Eingabe mit COMMIT TRANSACTION in die Tabelle TAUTOMAT eingelesen, und im Block AUTOMAT erscheinen:

$$T = 1, A = 0, AALT = 0 \text{ und } ANEU = 1$$

Wird UPDATE angefordert, ohne daß vorher EXECUTE QUERY durchgeführt wurde, befindet man sich nicht im UPDATE- sondern im INSERT-Modus. Werden jetzt die Werte bei T oder A geändert, wird COMMIT TRANSACTION zurückgewiesen. Dies geschieht durch das Attribut R, das Primärschlüsselfunktion hat, für das aber in der Form keine Eingaben etc. erfolgen können. Dadurch ist sichergestellt, daß beim Arbeiten mit der Form kein zweites Tupel in die Relation TAUTOMAT eingefügt werden kann.

---

31) Warum diese Meldung erscheint, ist ein echtes ORACLE-Geheimnis, das ebenfalls auf der ORACLE-Anwendertagung angesprochen wurde und einer Lösung harrt.

Enthalten der Block AUTOMAT und die Relation TAUTOMAT die gewünschten Kennziffern, wechselt man wieder in den Block TALK, wobei der Cursor auf das Feld SPRECHER springt. Im Block TALK müssen auf jeden Fall Einträge in den Feldern SPRECHER und WORTLAUT vorgenommen werden, um beim Beispiel zu bleiben, den Sprecher und den Wortlaut der ersten Äußerung (ANEU = 1), die aus dem Text mit TNR = 1 stammt. Mit COMMIT TRANSACTION kann dieser Eintrag in die Relation TALK eingelesen werden. Die Werte, die bei T und ANEU angezeigt werden, werden als Werte für die Felder TNR und ANR im Block TALK eingelesen, in die Relation TALK übernommen und in den beiden Feldern im Block TALK angezeigt. Fordert man jetzt für die Eingabe NEXT RECORD an, werden alle Einträge im Block TALK gelöscht, der Cursor springt wieder auf das Feld SPRECHER, und im Block AUTOMAT wird der Wert AALT auf "1" gesetzt und ANEU auf "2". Die übrigen Werte im Block AUTOMAT (T = 1, A = 0) werden nicht verändert.

Enthält die Relation TALK bereits Einträge, dann enthält die Tabelle TAUTOMAT die Werte von TNR und ANR der letzten Eingabe (z.B. 5,27) in die Relation TALK. Diese werden im Block AUTOMAT angezeigt, wenn man dort mit EXECUT QUERY eine Abfrage durchführt, d.h. man erhält im Block AUTOMAT dann:

T = 5, A = 27, AALT = 27 und ANEU = 28.

Will man die Eingabe für diesen Text fortsetzen, geht man ohne UPDATE wieder in den Block TALK und arbeitet mit Äußerung "28" weiter. Will man einen anderen Text bearbeiten, der noch nicht vollständig eingegeben ist, führt man nach EXECUTE QUERY zunächst wieder ein UPDATE durch, indem man für T die entsprechende Textnummer eingibt und für A die Äußerungsnummer der zuletzt eingegebenen Äußerung dieses Textes. Beginnt man mit der ersten Äußerung eines Textes, dann ist beim UPDATE die Textnummer einzugeben und als Äußerungsnummer der Wert "0".

Der Block AUTOMAT ist so ausgelegt, daß pro Text der Wert für TNR nur einmal eingegeben werden muß (in das Feld T) und danach automatisch jeder Äußerung des betreffenden Textes zugewiesen wird. Für ANR ist die Startnummer minus "1" einzugeben. Ausgehend von diesem Wert wird solange automatisch hochgezählt, bis der Wert wieder über UPDATE verändert wird.

Die Eingabe in die Relation TALK erfolgt über den Block TALK. Sie beginnt mit dem Feld SPRECHER. Die Eingabe kann hier und bei den übrigen Feldern frei erfolgen. Bei den Feldern SPRECHER, AGENS, POSFINITUM, ILLOKUTION, DISKURSFUNKTION und FUNKTION können die Werte wieder über LIST OF VALUES abgerufen und eingegeben werden. Die Listen befinden sich wieder in gesonderten Tabellen, analog zu den Hilfstabellen für d_entry. Diese Hilfstabellen werden, ebenfalls analog zu d_entry, zur Prüfung, ob die Eingabe formal korrekt ist, herangezogen, weshalb es z.B. eine Hilfstabelle

HILFE_AG(AGENS) mit den Werten "+" und "-" gibt. Die Prüfung wird durch den bereits erwähnten POST-CHANGE Trigger geleistet.

Das Feld WORTANZAHL, das uns bereits Probleme sehr grundsätzlicher Art beschert hat (vgl. Abschnitt 5.3.3), erweist sich bei der Entwicklung auf Formebene als pflegeleicht oder anderes ausgedrückt, hier ist weder etwas anzubieten noch abzusichern. Pro Äußerung sind halt die Wortformen zu zählen und einzugeben.

Für das Feld WORTLAUT gibt es derzeit keine online-Hilfe, außer dem bescheidenen Hinweis, daß die Hilfen dem Manual zu entnehmen sind, denn mit den im Abschnitt 6.6 bereits dargelegten Problemen geht einher, daß die Hilfen höchst idiosynkratischer Art und recht umfangreich sind, weshalb wir auf ihre Darstellung verzichten. Wir weisen lediglich darauf hin, daß bei diesem Feld mit SCROLL gearbeitet werden muß. Bei der Eingabe sollte dies in dem durch | PF22 | gekennzeichneten Bereich erfolgen. Sind Eingaben bei den Feldern SPRECHER und WORTLAUT erfolgt, dann kann die Information in die Relation TALK eingelesen werden.

c) Triggers und Hilfen

Da die Hilfen in gleicher Weise wie bei d_entry strukturiert sind, gehen wir hier nur noch auf die Triggers ein. Beim Arbeiten mit der Form t_entry wirken insgesamt 3 verschiedene Triggertypen mit:

Eingesetzt wurde wieder der bereits bekannte POST-CHANGE Trigger, der beim Feld T im Block AUTOMAT greift und zur Aufrechterhaltung der referentiellen Integrität beiträgt:

```
SELECT 'X'
FROM TEXTE
WHERE TEXTE.TNR = :TAUTOMAT.T
```

Außerdem wurde der POST-CHANGE Trigger im Block TALK wieder zur Validierung der Eingabe (s.o.) verwendet.

Beim Feld SPRECHER im Block TALK wurde ein PRE-FIELD Trigger gesetzt, der aus zwei Schritten besteht:

```
SEQ# 1
SELECT A
INTO TAUTOMAT.AALT[32)]
FROM TAUTOMAT
```

```
SEQ# 2
SELECT A+1
INTO TAUTOMAT.ANEU
FROM TAUTOMAT
```

---

32) Der Block AUTOMAT hat intern den gleichen Namen wie die Relation, also TAUTOMAT und wird lediglich auf der Bildschirmausgabe als AUTOMAT bezeichnet. In dem INTO-Statement wird durch TAUTOMAT der Block bezeichnet, im nachfolgenden FROM-Statement die Tabelle.

Wenn der Cursor auf das Feld SPRECHER springt, wird der Trigger gestartet. Das ist der Fall, wenn die Form aufgerufen wird, wenn man vom Block AUTOMAT in den Block TALK wechselt, wenn man vom Feld WORTLAUT mit PREVIOUS FIELD wieder auf das Feld zurückgeht oder NEXT RECORD im Block TALK anfordert. Aus der Tabelle TAUTOMAT wird der Wert für A in das Feld AALT im Block AUTOMAT eingelesen. Im zweiten Schritt wird "1" zum Wert "a" addiert und dann in das Feld ANEU eingelesen.

Während die beiden ersten Trigger auf Feldebene wirken, arbeitet der dritte, ein PRE-INSERT-Trigger, auf Blockebene und wird durch COMMIT TRANSACTION gestartet. Der Trigger besteht aus drei Schritten:

```
SEQ# 1                SEQ# 2              SEQ# 3
SELECT T              UPDATE TAUTOMAT     SELECT A
INTO TALK.TNR         SET A = A+1         INTO TALK.ANR
FROM TAUTOMAT                             FROM TAUTOMAT
```

Zunächst wird aus der Tabelle TAUTOMAT der aktuelle Wert von T für das Feld TNR in den Block TALK übernommen (SEQ# 1). Außerdem erfolgt ein UPDATE für den Wert von A; der aktuelle Wert "a" wird um "1" hochgesetzt und mit diesem neuen Wert "a+1" überschrieben (SEQ# 2). Der dritten Schritt (SEQ# 3) leistet nun für ANR im Block TALK dasselbe wie der erste Schritt für TNR. Erst wenn der Trigger vollständig abgearbeitet ist, werden die Einträge in die Relation TALK eingelesen, und im Block TALK werden die Werte für TNR und ANR angezeigt. Eine grundlegende Voraussetzung für das fehlerfreie Zusammenspiel zwischen den T- bzw. A- und TNR- bzw. ANR-Werten ist in TAUTOMAT enthalten. Die Relation darf nur ein Tupel enthalten. Wenn das nicht der Fall ist, erfolgen falsche Zuordnungen. Aus diesem Grund wurde das Attribut R mit Primärschlüsselfunktion eingefügt und dafür gesorgt, daß auf Formebene keine Eingaben in diesem Feld möglich sind.

Um Mißverständnisse zu vermeiden: (a) Eingaben in das Feld R sind auf SQL*Plus-Ebene möglich, nicht jedoch auf Form-Ebene. (b) Der PRE-FIELD Trigger ändert den Eintrag in der Tabelle TAUTOMAT nicht, er sorgt lediglich dafür, daß die Äußerungsnummer der zuletzt an die Relation TALK abgegebenen Äußerung angezeigt wird sowie die Nummer der nächsten einzugebenden Äußerung. In Hinblick auf die Relation TALK wird die Äußerungsnummer erst durch das UPDATE von A in der Tabelle TAUTOMAT bereitgestellt. (c) An diesem Prozeß ändert sich auch dann nichts, wenn man mehrfach NEXT RECORD anfordert, ohne vor jeder neuen Anforderung COMMIT TRANSACTION ausgeführt zu haben. Der PRE-INSERT Trigger wird, um es salopp auszudrücken, pro einzulesendem *record* abgearbeitet. Was allerdings in diesem Fall passiert, ist folgendes: Die Werte von AALT und ANEU verändern sich nicht, da in TAUTOMAT kein UPDATE vorgenommen wird, wenn NEXT RECORD angefordert wird, sondern dies nur im Zusammenhang mit COMMIT TRANSACTION geschieht. Es empfiehlt sich also, die *records* einzeln und nicht gesammelt an die Relation TALK abzugeben.

7.4. p_entry und PHRASEN

Die Form p_entry für PHRASEN (Abbildung 7) besteht aus den Blöcken AUTOMAT und PHRASEN und umfaßt eine Bildschirmseite. Ebenso wie bei d_entry und t_entry reicht der Platz im Block PHRASEN nur für ein *record.*

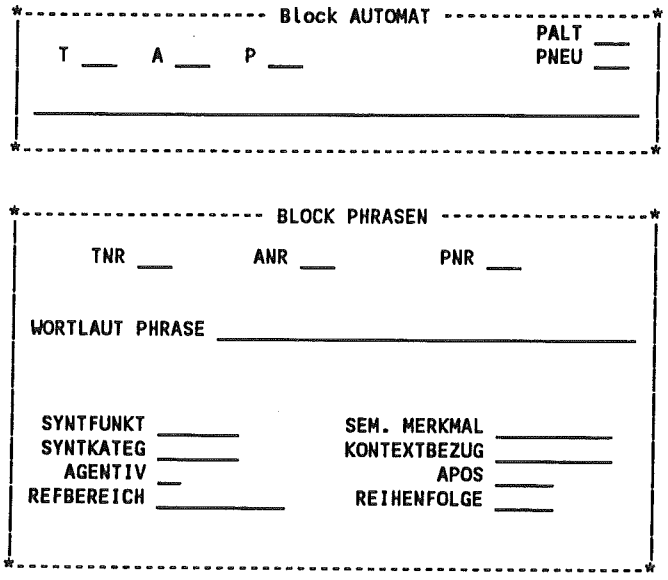

Abbildung 7: *Bildschirmausgabe der Form p_entry*

a) Beschreibung der Felder

Die Felder T, A und P im Block AUTOMAT (Seq# 2) gehören zu der Tabelle PAUTOMAT, die als weiteres Attribut R enthält. Ansonsten gelten die gleichen Ausführungen wie für den Block AUTOMAT der Form t_entry. Die Felder PALT und PNEU dienen zur Darstellung der Phrasennummern pro Äußerung. Außerdem enthält der Block als weiteres Feld WORTLAUT_AEUS. Es ist ebenso wie PALT bzw. PNEU kein DATABASE FIELD und ist nur als DISPLAYED spezifiziert. In dieses Feld wird der Wortlaut der Äußerung, die in Phrasen segmentiert werden soll, eingelesen. Aus Platzersparnisgründen wurde für dieses Feld keine Bezeichnung auf der Bildschirmausgabe angegeben.

Die Felder im Block PHRASEN (Seq# 1) sind von WORTLAUT_PHR (Seq# 1) bis REIHENFOLGE (Seq# 9) zeilenweise durchnumeriert; TNR, ANR und PNR haben die Nummern Seq# 10, Seq# 11 und Seq# 12. Die Felder gehören alle zur Relation PHRASEN, in der die Felder TNR, ANR, PNR und WORTLAUT_PHR als "NOT NULL", TNR, ANR und PNR als NUMBER definiert sind und die übrigen Felder als CHAR. Auf Form-Ebene wurden für TNR, ANR und PNR nur die Spezifikationen DATABASE

FIELD und QUERY ALLOWED beibehalten, NUMBER wurde wieder in RINT geändert. Bei den CHAR-Feldern wurde UPPERCASE spezifiziert, ausgenommen sind hiervon WORTLAUT_PHR und APOS.

b) Funktionsweise

Die Funktionsweise von p_entry folgt den gleichen Prinzipien wie t_entry. Durch den Block AUTOMAT wird die Bearbeitung des Blocks PHRASEN unterstützt. Die verwendeten Triggertypen sind - bis auf einen POST-BLOCK Trigger - die gleichen wie bei t_entry: PRE-FIELD, und PRE-INSERT. Hilfen und Validierung der Eingaben über LIST OF VALUES, die wiederum in entsprechenden Hilfstabellen sind, werden genauso gehandhabt wie bisher. Solche Hilfstabellen bestehen für die Attribute SYNTFUNKT, SYNTKATEG, AGENTIV, REFBEREICH, SEMMERKMAL, KONTEXTBEZUG und REIHENFOLGE. Vergleichbare Hilfstabellen gibt es weder für WORTLAUT_PHR noch für APOS, da die Werte hier stark variieren.

c. Triggers und Hilfen

Der PRE-FIELD Trigger, der PALT und PNEU regelt, ist beim Feld WORTLAUT_PHR des Blocks PHRASEN gesetzt. Er besteht wieder aus zwei Schritten, deren Funktionsweise bereits dargestellt wurde:

```
SEQ# 1                          SEQ# 2
SELECT P                        SELECT P+1
INTO PAUTOMAT.PALT              INTO PAUTOMAT.PNEU
FROM PAUTOMAT                   FROM PAUTOMAT
```

Ein POST_BLOCK Trigger ist beim Block AUTOMAT gesetzt. Er dient zum einen dazu, die referentielle Integrität zu prüfen, und zum anderen dazu, aus der Relation TALK die Äußerung einzulesen, die in Phrasen segmentiert werden soll. Er besteht aus zwei Schritten:

```
SEQ# 1                          SEQ# 2
SELECT 'X'                      SELECT WORTLAUT
FROM TALK                       INTO PAUTOMAT.WORTLAUT_AUES
WHERE TALK.TNR = :PAUTOMAT.T    FROM TALK
 AND TALK.ANR = :PAUTOMAT:A     WHERE TALK.TNR = :PAUTOMAT.T
                                 AND TALK.ANR = :PAUTOMAT.A
```

Wenn in den Feldern T bzw. A ein UPDATE vorgenommen wird, was immer der Fall ist, wenn eine neue Äußerung in Phrasen segmentiert werden soll, wird über den ersten Schritte geprüft, ob eine derartige Kombination überhaupt vorhanden ist. Ist das nicht der Fall, wird die Fehlermeldung "Die gewünschte Äußerung ist noch nicht eingetragen. Bitte nachholen oder eine andere bearbeiten." ausgegeben. Die Aufforderung kann nicht umgangen werden, da man aus dem Block AUTOMAT nicht rauskommt, ohne eine bereits in der Relation TALK vorhandene TNR,ANR-Kombination einzugeben. Wählt man eine dort vorhandene Kombination, deren Äußerung auf Phrasenebene bereits vollständig bearbeitet wurde, können die nachfolgend im Block PHRASEN vorgenommenen Einträge nicht in die Relation TALK eingelesen

werden, da dort die TNR,ANR,PNR-Werte bereits vorhanden sind, die ja den Primärschlüssel bilden. Den Block AUTOMAT umgehen zu wollen, macht schon allein deshalb keinen Sinn, weil im Block PHRASEN TNR, ANR und PNR während der Eingabe nicht direkt vergeben werden können.

Der letzte Trigger, PRE-INSERT entspricht dem bei t_entry, d.h. nimmt ein UPDATE in PAUTOMAT vor und versieht die Eingabe mit den Primärschlüsselwerten. Er besteht aus vier Schritten:

```
SEQ# 1
SELECT T
INTO PHRASEN.TNR
FROM PAUTOMAT
```

```
SEQ# 2
SELECT A
INTO PHRASEN.ANR
FROM PAUTOMAT
```

```
SEQ# 3
UPDATE PAUTOMAT
SET P = P+1
```

```
SEQ# 4
SELECT P
INTO PHRASEN.PNR
FROM PAUTOMAT
```

Ein letzter Punkt bleibt zu erwähnen:

Wir haben derzeit keine halbwegs passable datenbankinterne Lösung für den Fall, daß die Datenbank einen Wortlaut enthält, der eigentlich zwei Äußerungen umfaßt. Dies deshalb nicht, weil in unserer Anwendung die Reihenfolge der Äußerungen im Text berücksichtigt werden muß. Das heißt natürlich nicht, daß das Tupel für die zweite Äußerung nicht vor dem Tupel für die erste stehen darf. Entscheidend ist, daß die Äußerungsnummern, die Bestandteil des Primärschlüssel sind, die Reihenfolge der Äußerungen im Text abbilden müssen, und hier liegt das Problem:

Wenn der Wortlaut, der zu ANR = 1 gehört irrtümlich zwei Äußerungen umfaßt, aber ein anderer Wortlaut bereits mit ANR = 2 versehen ist, erweisen sich die datenbankinternen UPDATE-Möglichkeiten als ausgesprochen schwerfällig; man könnte z.B. die letzte Äußerungnummer um die Anzahl der Äußerungen, die eingefügt werden müssen, hochsetzen, dann die vorletzte usw. Dies ist die umständlichste Art, aber auch die anderen Möglichkeiten sind immer noch recht aufwendig, da sie das Anlegen neuer Relationen als "Umschaufeln der Daten" usw. erfordern. Eine Alternative zu diesen datenbankinternen Verfahren besteht darin, den gesamte Text zu exportieren, zu korrigieren und wieder einzulesen. Das allein genügt jedoch nicht, weil das Problem auf die Relation PHRASEN "vererbt" wird, und auch dort gilt dann: Exportieren, korrigieren, importieren. Es gilt aber auch, daß in solchen Fällen nie der gesamte Inhalt einer Relation exportiert werden muß, sondern nur der betreffende Text.

Im Zusammenhang damit stellt sich allemal die Frage, ob es nicht grundsätzlich sinnvoller ist, die klassifizierten Äußerungen z.B. in einer ASCII-Datei festzuhalten (vgl. S. 41), dort bereits mit den Werten für TNR und ANR zu versehen und die Datei erst dann in die Datenbank einzuspielen, wenn sämtliche Kor-

rekturen vorgenommen sind. In gleicher Weise könnte für den Wortlaut der Phrasen verfahren werden. So vorzugehen empfiehlt sich schon allein deshalb, weil die Bearbeitung der Texte über Texteditoren wesentlich weniger zeitintensiv ist als über die Datenbank, was auf jeden Fall für die Korrektur von "Tippfehlern" gilt, aber auch für das Ändern der Schlüsselwerte. Nicht zuletzt könnte dann datenbankintern auf die schwerfälligen Zählautomaten verzichtet werden.

8. Zu Nutz und Frommen

Die folgenden Sätze fassen die Erfahrungen zusammen, die wir für typisch und gewichtig halten. Sie mögen von allen bedacht werden, die keine Erfahrungen mit Datenbankanwendungen haben und erwägen, komplexe Sachverhalte systematisch empirisch zu analysieren, seien dies sprachliche Äußerungen, Noten, historische Dokumente, Bilder, Sozial- und Individualdaten oder Kombinationen davon.

1. Die Frage nach dem Einsatz oder Nicht-Einsatz eines Datenbankprogramms ist im Kern dieselbe wie die, ob man sich beim Analysieren seiner Belege Notizen machen soll oder nicht; lediglich das Verfahren ist aufwendiger, aber auch effizienter.

2. Der Einsatz eines Datenbankinstruments setzt gründliche und dauerhaft verfügbare Sachkenntnisse des Datenbankprogrammpakets, der Besonderheiten der verwendeten Rechner und der Projektfragestellung selbst voraus.

3. Auch wenn die Erwägungen zu den Punkten 1 und 2 den Einsatz eines Datenbanksystems angezeigt und möglich erscheinen lassen, ist ein entsprechender Entschluß noch ein besonderer Schritt, der leicht verschoben wird.

4. Die Arbeit, die für die Planung der Datenbank getan werden muß, ist schon Arbeit an der Sache selbst.

5. Wenn die Forderungen aus Punkt 2 in vollem Umfang erfüllt sind, was in vergleichbaren Projekten eher die Ausnahme als die Regel ist, dann erübrigen sich die Punkte 6 bis 10. Wenn nicht, dann mögen auch sie bedacht werden.

6. Man beschränke sich bei der Anwendungsentwicklung auf ein Minimum an Komplexität, so daß die entwickelte Anwendung überschaubar und durch Nicht-Fachleute übernehmbar bleibt, auch wenn dies Datenbankspezialisten gegen den Strich gehen mag.

7. Die frühzeitigen Rezeption nicht nur der Handbücher des jeweiligen Programmpakets, sondern insbesondere der grundlegenden programmunabhängigen Literatur kann vor gravierenden Fehleinschätzungen- und leistungen bewahren. Die hierfür verwendete Zeit zahlt sich allemal aus.

8. Konzeptionelle Planung, Anwendungsentwicklung und Implementierung sollten nicht, wie häufig empfohlen, sequentiell sondern interagierend erfolgen.

9. Lösungen für Probleme, die in der einschlägigen Datenbankliteratur nicht diskutiert werden, weil sie sachverhaltsspezifisch sind, sollten in ihren Auswirkungen immer wieder geprüft werden. Es könnte sich auf lange Sicht (mit zunehmendem Umfang der Datenbank) erweisen, daß man den Teufel mit dem Beelzebub ausgetrieben hat (hier sei an TAUTOMAT und PAUTOMAT erinnert, die die online-Eingabe erleichtern sollen).

10. Häufig wird empfohlen, Forms endanwenderspezifisch zu entwickeln, wobei unterstellt wird, daß der Endanwender am besten nichts über die Datenbank wissen sollen müßte. Das mag in den üblichen kommerziellen Anwendungen der Fall und vielleicht wünschenswert sein, in vergleichbaren Projekten halten wir dies eher für kontraproduktiv.

11. "DON'T PANIC!"[33)]

33) Ein notorischer Rat aus "The Hitchhiker's Guide to the Galaxy" von Douglas Adams.

LITERATUR

Altmann, H. (1987). Formen der Herausstellung im Deutschen. Tübingen: Niemeyer.

Behagel, O. (1932). Deutsche Syntax. Eine geschichtliche Darstellung. Bd. IV: Wortstellung, Periodenbau. Heidelberg: Carl Winters Universitätsbuchhandlung.

Boost, K. (1957). Neue Untersuchungen zum Wesen und zur Struktur des deutschen Satzes. Der Satz als Spannungsfeld. Berlin: Deutsche Akademie der Wissenschaften. Institut für deutsche Sprache und Literatur. 1. Aufl. 1955.

Chen, P.P. (1976). The Entity-Relationship Model - Toward a Unified View of Data. ACM TODS 1:1.

Codd, E. F. (1969). Derivability, Redundancy, and Consistency of Relations Stored in Large Data Banks. San Jose, Cal.: IBM Research Report RJ559.

Codd, E. F. (1990). The Relational Model for Database Management: Version 2. Reading, Mass.: Addison-Wesley.

Date, C. J. (1990). An Introduction to Database Systems. Vol. 1, 5th ed. Reading, Mass.: Addison-Wesley.

Diemer, W. R. (1989). Relationale Datenbanken kurz und bündig.

Dietrich, R. & Schneider, W. (1989). Designing computer tools in the humanities as non-standard database toolsystems. CAK, 7, 1-12.

Dietrich, R. & zu Pulitz, G. (1988). Sicher, aber langsam. Die Zukunft der Datenverarbeitung in den Geisteswissenschaften. Deutsche Universitätszeitung, 22, 19-22.

Dowty, R. (1989). On the semantic content of the notion "thematic role". In: G. Chierchia, B. H. Partee & R. Turner (Eds.): Properties, Type and Meaning. Vol. 2, pp. 69-130. Dordrecht: Kluwer Academic Press.

Gillner, R. (1984). Datenbanken auf Arbeitsplatzrechnern. München/Wien: Carl Hanser Verlag.

Herrmann, Th. (1985). Allgemeine Sprachpsychologie. München: Urban & Schwarzenberg.

Levelt, W.J.M. (1989). Speaking: From Intention to Articulation. Cambridge, Mass.: The M.I.T.-Press.

Martin, J. (1987). Einführung in die Datenbanktechnik. München/Wien: Carl Hanser Verlag.

Reichel, W. (1987). Die deutsche Wortstellung in der Gegenwart. In: W. Reichel (Hrsg.): Sprachpsychologische Studien. S. 1-95. Halle a.S.: Max Niemeyer.

Ritter, J. (1974). Die Aufgabe der Geisteswissenschaften in der modernen Gesellschaft. In: J. Ritter (Hrsg.) Subjektivität. Frankfurt a.M.: Suhrkamp.

Rolland, F. D. (1990). Relational Database Management with ORACLE. Wokingham: Addison-Wesley.

Vetter, M. (1990). Aufbau betrieblicher Informationssysteme mittels konzeptioneller Datenmodellierung. 6. neubearb. u. erw. Auflg. Stuttgart: B. G. Teubner.

V E R Z E I C H N I S

der Arbeiten aus dem Sonderforschungsbereich 245

"Sprechen und Sprachverstehen im sozialen Kontext"

Heidelberg/Mannheim

Nr. 1 Schwarz, S., Wagner, F. & Kruse, L.: Soziale Repräsentation und Sprache: Gruppenspezifische Wissensbestände und ihre Wirkung bei der sprachlichen Konstruktion und Rekonstruktion geschlechtstypischer Episoden. Februar 1989.

Nr. 2 Wintermantel, M., Laux, H. & Fehr, U.: Anweisung zum Handeln: Bilder oder Wörter. März 1989.

Nr. 3 Herrmann, Th., Dittrich, S., Hornung-Linkenheil, A., Graf, R. & Egel, H.: Sprecherziele und Lokalisationssequenzen: Über die antizipatorische Aktivierung von Wie-Schemata. April 1989.

Nr. 4 Schwarz, S., Weniger, G. & Kruse, L. (unter Mitarbeit von R. Kohl): Soziale Repräsentation und Sprache: Männertypen: Überindividuelle Wissensbestände und individuelle Kognitionen. Juni 1989.

Nr. 5 Wagner, F., Theobald, H., Heß, K., Schwarz, S. & Kruse, L.: Soziale Repräsentation zum Mann: Gruppenspezifische Salienz und Strukturierung von Männertypen. Juni 1989.

Nr. 6 Schwarz, S. & Kruse, L.: Soziale Repräsentation und Sprache: Gruppenspezifische Unterschiede bei der sprachlichen Realisierung geschlechtstypischer Episoden. Juni 1989.

Nr. 7 Dorn-Mahler, H., Grabowski-Gellert, J., Funk-Müldner, K. & Winterhoff-Spurk, P.: Intonation bei Aufforderungen. Teil 1: Theoretische Grundlagen. Juni 1989.

Nr. 8 Dorn-Mahler, H., Grabowski-Gellert, J., Funk-Müldner, K. & Winterhoff-Spurk, P.: Intonation bei Aufforderungen. Teil II: Eine experimentelle Untersuchung. Dezember 1989.

Nr. 9 Sommer, C.M. & Graumann, C.F.: Perspektivität und Sprache: Zur Rolle von habituellen Perspektiven. August 1989.

Nr. 10 Grabowski-Gellert, J. & Winterhoff-Spurk, P.: Schreiben ist Silber, Reden ist Gold. August 1989.

Nr. 11 Graf, R. & Herrmann, Th.: Zur sekundären Raumreferenz: Gegenüberobjekte bei nicht-kanonischer Betrachterposition. Dezember 1989.

Nr. 12 Grosser, Ch. & Mangold-Allwinn, R.: Objektbenennung in Serie: Zur partnerorientierten Ausführlichkeit von Erst- und Folgebenennungen. Dezember 1989.

Nr. 13 Grosser, Ch. & Mangold-Allwinn, R.: Zur Variabilität von Objektbenennungen in Abhängigkeit von Sprecherzielen und kognitiver Kompetenz des Partners. Dezember 1989.

Nr. 14 Gutfleisch-Rieck, I., Klein, W., Speck, A. & Spranz-Fogasy, Th.: Transkriptionsvereinbarungen für den Sonderforschungsbereich 245 "Sprechen und Sprachverstehen im sozialen Kontext". Dezember 1989.

Nr. 15 Herrmann, Th.: Vor, hinter, rechts und links: das 6H-Modell. Psychologische Studien zum sprachlichen Lokalisieren. Dezember 1989.

Nr. 16 Dittrich, S. & Herrmann, Th.: "Der Dom steht hinter dem Fahrrad." - Intendiertes Objekt oder Relatum? März 1990.

Nr. 17 Kilian, E., Herrmann, Th., Dittrich, S. & Dreyer, P.: Was- und Wie-Schemata beim Erzählen. Mai 1990.

Nr. 18 Herrmann, Th. & Graf, R.: Ein dualer Rechts-links-Effekt. Kognitiver Aufwand und Rotationswinkel bei intrinsischer Rechts-links-Lokalisation. August 1990.

Nr. 19 Wintermantel, M.: Dialogue between expert and novice: On differences in knowledge and means to reduce them. August 1990.

Nr. 20 Graumann, C.F.: Perspectivity in Language and Language Use. September 1990.

Nr. 21 Graumann, C.F.: Perspectival Structure and Dynamics in Dialogues. September 1990.

Nr. 22 Hofer, M., Pikowsky, B., Spranz-Fogasy, Th. & Fleischmann, Th.: Mannheimer Argumentations-KategorienSystem (MAKS). Mannheimer Kategoriensystem für die Auswertung von Argumentationen in Gesprächen zwischen Müttern und jugendlichen Töchtern. Oktober 1990.

Nr. 23 Wagner, F., Huerkamp, M., Jockisch, H. & Graumann, C.F.: Sprachlich realisierte soziale Diskriminierungen: empirische Überprüfung eines Modells expliziter Diskriminierung. Oktober 1990.

Nr. 24 Rettig, H., Kiefer, L., Sommer, C.M. & Graumann, C.F.: Perspektivität und soziales Urteil: Wenn Versuchspersonen ihre Bezugsskalen selbst konstruieren. November 1990.

Nr. 25 Kiefer, L., Sommer, C.M. & Graumann, C.F.: Perspektivität und soziales Urteil: Klassische Urteilseffekte bei individueller Skalenkonstruktion. November 1990.

Nr. 26 Hofer, M., Pikowsky, B., Fleischmann, Th. & Spranz-Fogasy, Th.: Argumentationssequenzen in Konfliktgesprächen zwischen Müttern und Töchtern. November 1990.

Nr. 27 Funk-Müldner, K., Dorn-Mahler, H. & Winterhoff-Spurk, P.: Kategoriensystem zur Situationsabhängigkeit von Aufforderungen im betrieblichen Kontext. Dezember 1990.

Nr. 28 Groeben, N., Schreier, M. & Christmann, U.: Argumentationsintegrität (I): Herleitung, Explikation und Binnenstrukturierung des Konstrukts. Dezember 1990.

Nr. 29 Blickle, G. & Groeben, N.: Argumentationsintegrität (II): Zur psychologischen Realität des subjektiven Wertkonzepts - ein experimenteller Überprüfungsansatz am Beispiel ausgewählter Standards. Dezember 1990.

Nr. 30 Schreier, M. & Groeben, N.: Argumentationsintegrität (III): Rhetorische Strategien und Integritätsstandards. Dezember 1990.

Nr. 31 Sachtleber, S. & Schreier, M.: Argumentationsintegrität (IV): Sprachliche Manifestationen argumentativer Unintegrität - ein pragmalinguistisches Beschreibungsmodell und seine Anwendung. Dezember 1990.

Nr. 32 Dietrich, R., Egel, H., Maier-Schicht, M. & Neubauer, M.: ORACLE und die Analyse des Äußerungsaufbaus. Februar 1991.