TECHNISCHE UNIVERSITAET WIEN

Diese Arbeit hat beurteilt:

DIPLOMARBEIT

Thema: Analyse von GERM

ausgeführt am Institut fuer Praktische Informatik -
Abteilung fuer Programmiersprachen und Übersetzerbau

der Technischen Universität Wien

unter Anleitung von Univ.Ass. Dipl.Ing. Franz REICHL
und Univ.Prof. BROCKHAUS

durch HERTEL Nicole

Theodor-Sickelgasse 16-20/9/6
1100 WIEN

Wien, am 9.Jän. 1986

Hertel Nicole
Unterschrift

Der Firma E. Schrack danke ich für die Betreuung und für die Bereitstellung des erforderlichen Materials.

ANALYSE VOM GERM

N. HERTEL

880 8225076

Die Analyse vom GERM erfolgt einerseits an Hand eines Kennzeichenkataloges fuer relationale Datenbanken, andererseits an Hand eines Beispieles einer Implementierung.

9. Januar 1986

34 Seiten

1 ALLGEMEINES

1.1 AUFGABENSTELLUNG

In diesem Dokument wird Germ, eine Datenbanksprache fuer eine relationale Datenbank, auf zwei Arten untersucht:

- einerseits an Hand eines Kennzeichen= kataloges, der zur Analyse der relationaler Datenbanken von Relationale Task Group - RTG (ANSI) entwickelt wurde.

- andererseits an Hand eines Beispiels einer Implementierung.

1.2 VORWORT

Hoehere Programmiersprachen wie FORTRAN, COBOL oder auch REXX oeffneten wesentlich mehr Fachleuten den Zugriff zu Datenverarbeitungsanlagen als es vorher unter Verwendung von Maschinensprachen oder Assemblersprachen der Fall war. Aber auch die Anschluss= moeglichkeit entsprechend an die einzelnen Benutzer "angepassten" Datenverarbeitungsanlagen veraenderte die Situation in der gesamten Computerbranche erheblich.

Ein beinahe ebenso revolutionaer erscheinender Schritt erfolgte durch die Verwendung von allgemeinen Datenbank= systemen und innerhalb der unterschiedlichen Datenbanksystemen durch die Einfuehrung der relationalen Datenbanken. Ein vor allem fuer die Privatwirtschaft sehr grosser Vorteil besteht darin, dass Anwendungsprobleme, zu deren Loesung vorher selbst erfahrene Programmierer bei Verwendung von Standard-Programmiersprachen einen Zeitraum von mehren Wochen benoetigten,nun durch die Einfuehrung der relationalen Datenbanken selbst von EDV-Unkundigen mit relativ geringem Zeitaufwand selbstaendig geloest werden koennen.

Datenbanksysteme unterstuetzen die Erfassung, Aenderung von Daten, die fuer den einzelnen Benutzer von grosser Bedeutung sind.

Die Benutzung von relationalen Datenbanksystemen eroeffnet auch dem EDV-Laien zum erstenmal in der Geschichte den direkten Zugriff zum Rechner.

2 RELATIONALE DATENBANKEN

2.1 VORTEILE DER RELATIONALEN DATENBANKEN

Ein wesentlicher Vorteil der relationalen Datenbanken gegenueber anderen - wie zum Beispiel den hierarchischen Datenbanksystemen - liegt darin, dass keine Zugriffspfade von vorhinein definiert werden muessen.

Die Produktivitaet des Datenbanksystemes wurde um einen Faktor 5 bis 20 erhoeht.

Diese Erhoehung findet ihre Begruendung darin, dass sich der Endbenutzer oder Anwender selbst Tabellen kreieren kann, deren Elemente genau seinen Vorstellungen entsprechen; so kann er sich aus der Datenbank mit einfachen und leicht erlernbaren Befehlen alle Daten aus der gespeicherten Datenbank holen, die seiner Anwendung entsprechen.

Somit kann sich jeder Benutzer, ohne dass vom Anwendungsprogrammierer vorher ein eigenes Programm geschrieben werden muesste, seine Information selbsttaetig und einfach aus der Datenbank holen.
Dadurch findet also bei der Vielzahl der Endbenutzer und deren unterschiedlichen Anwendungen und gewuenschten Programmauswertungsergebnisse eine relationale Datenbank sicherlich ihre Bestaetigung.

2.2 CHARAKTERISTIKEN EINER RELATIONALEN DB

Die **Relationale Task Gruppe (R T G)** des Amerikanischen Nationalen Standards Institutes / **ANSI** / unternahm grosse Anstrengungen zur Entwicklung eigener charakteristischer Eigenschaften von relationalen Datenbanken.

2.2.1 MINIMALE KONDITIONEN

- Jede Information der Datenbank wird als Wert in einer Tabelle dargestellt.
  D.h.: Alle Information werden in Tabellen gespeichert;
  d.h.: eine relationale Datenbank ist **tabellenorientiert**

- Es gibt **keine** fuer den Benutzer sichtbaren verbindenden Links zwischen den

einzelnen Tabellen.

- .) Das System enthaelt zumindest die **Selektion, Projektion und den Join der relationalen Algebra** ; wobei die Form der Implementierung egal ist.

Wenn ein Datenbanksystem nur den oben genannten Anforderungenm entspricht, so nennt man sie **minimal relational**.

Ein Datenbanksystem, welche nur die beiden ersten oben angefuehrten Bedingungen erfuellt, nennt man **tabular**.

Ein **voll relationales Datenbanksystem** muss neben den Bedingungen des minial relationalen Datenbank= systemes noch folgende Punkte erfuellen:

- .) Es muss allen Operatoren der relationalen Algebra genuegen, wobei es egal ist in welcher Syntax diese realisiert werden.
- .) Es muss den beiden generellen Integritaets= bedingungen des relationalen Modells genuegen (Datenintegritaet und Verweis= integritaet).

2.2.2 BEMERKUNGEN

Die meisten bestehenden relationalen Datenbanksystemen sind ein Kompromiss zwischen den minimal relationalen und den voll relationalen.
In den meisten Faellen muessen sie im Bereich der Integritaetsbedingungen ausgefeilt und verbessert werden.
Es besteht kein Zweifel mehr, dass dies bereits geschehen ist oder dass es in kurzer Zeit geschehen wird.

2.3 KENNZEICHEN EINER RELATIONALEN DB

2.3.1 Entstehung des Kennzeichenkataloges

Der Kennzeichenkatolog wurde von der **RTG** zum Zwecke der Selektion und Analyse der relationalen Datenbanken entwickelt.
Damit koennen alle Datenbanksysteme in gleicher Form beschrieben werden.

Auch die nicht relationalen Datenbanken koenne mit Hilfe des Kataloges beschrieben werden, damit man

- •) unterscheiden kann, ob ein Datenbanksytem relational ist oder nicht;
- •) den Standard der relationalen Datenbanken erhoehen kann;
- •) die Datenbanken, die selbst nicht relational sind, aber Aspekte von relationalen Datenbanken unterstuetzen, erkennen kann.

2.3.2 UNTERSCHEIDUNG DER KENNZEICHEN

Ein weiteres Anliegen war die Einteilung der Kennzeichen in verschiedene Gruppen:

1.) Eigenschaften, die in jedem System auftreten, das relational ist.

2.) Eigenschaften, die nicht in jedem Datenbanksystem auftreten, die aber rasch eingfuegt werden koennen.

3.) Eigenschaften, die als wichtig angesehen werden, jedoch ausser in der Literatur oder in Forschungsdatenbanken noch nicht implementiert wurden.

4.) Eigenschaften, die in manchen Systemen auftreten, aber nicht auf weitere Systeme ueberzugreifen scheinen.

5.) Eigenschaften, die die relationalen Systemen von anderen unterscheiden.

Eigenschaften der 1. und 2. Gruppe sind die Grundlage fuer jedes Datenbanksystem.

Eigenschaften der Gruppe 3 und 4 sind die Grundlage fuer zukuenftige Systeme.

Eigenschaften der Gruppe 5 dienen der Unterscheidung von relationalen Datenbanken zu anderen Systemen.

2.4 DATENBANKSPRACHEN

Einige relationale Datenbanksysteme unterstuetzen eine "Programmieruntersprache", die man auf 2 Arten verwenden kann:

- .) interaktiv im Onlinebetrieb
- .) eingebettet in ein Applikationsprogramm und somit in einer Wirtsprache (z.B. REXX oder PL/1 .....)

So eine Sprache nennt man dann Doppelmodesprache.

2.4.1 VORTEILE EINER DOPPELMODE-SPRACHE

bestehen darin, dass

- .) Applikationsprogramme unabhaengig vom Programm aus im Onlinemode getestet werden koennen.
- .) so eine Sprache auch die Kommunikation zwischen Programmierern, Benutzern und Datenbankadministratoren steigert.
- .) sie leicht erlernbar ist.

2.5 DER KENNZEICHENKATALOG

Im nun folgenden Kapitel wird beschrieben, was durch den Kennzeichenkatalog festgelegt wird.

2.5.1 EINFUEHRUNG

2.5.1.1 IDENTIFIKATION:

enthaelt den Namen des Systems (und die Version, die erhaeltlich ist) zusammen mit der Entstehungsgeschichte.

SYSTEM

enthaelt den Entwicklungsstand des Systems zusammen mit zukuenftige Releasezeitpunkte. Geplante Erweiterungen und die eventuellen Fertigungsdatuma koennen festgehalten werden.

APPLIKATIONEN

Die Klasse der Applikationen sollen hier angegeben werden; weiters die Charakterisierung der Grenzen, die vom System her bekannt sind z.B. der benoetigte Speicherplatz.

2.5.1.2 SYSTEMHINTERGRUND

Genaue Beschreibung der Entwicklung des System; zu welcher Familie von aehnlichen Systemen es gehoert und (wenn vorhanden) auf welchem System es aufbaut.

2.5.1.3 ZUGRUNDELIEGENDE PHILOSOPHIE

Beschreibung der zugrundeligenden Philosophie und der Motivitation der Entwicklung des Systems.

2.5.1.4 NOTWENDIGE RELATIONALE CHARAKTERISTIKEN

Codd schrieb dies fuer ein **voll relationales** Datenbanksystem:

Wobei fuer ein relationales Datenbanksystem folgende Eigenschaften vorhanden sein muessen.

- .) Strukturaspekte des relationalen Modells
- .) die INSERT, UPDATE und DELETE-Regeln
- .) eine Datenunterprogrammiersprache, die so maechtig, wie die relationale Algebra sein muss, auch wenn alle Moeglichkeiten der iterativen Schleifen und Rekursionen aus der Sprache entfernt wurden.

Codd definiert auch, dass eine relationale Datenbank, die keine solche "Datenunterprogrammiersprache" beinhaltet, **semirelational** genannt werden soll.

Der Standard der "Datenunterprogrammiersprache" ist von wesentlicher Bedeutung. Die Operatoren der relationalen Algebra, wie kartesisches Produkt, Vereinigung, Durchschnitt, Differenz, Projektion, Selektion, Auswahl, Join und vielleicht noch Division (Umkehrung zum kartesischen Produkt) muessen elementar sein.

Ein Datenbanksystem, welche Restriktionen bei der Verwendung der Operationen der relationalen Algebra auferlegt, wird **semirelational** genannt.

Nachfolgend werden einige solche Restriktionen zum Join angegeben:

- •) Kreise werden nicht behandelt.
- •) Eine Relation kann nicht mit sich selbst einen Join bilden.
- •) Ein Join behandelt nur 2 Relationen.
- •) Relationen muessen mittels Join verbunden sein.

Die nun folgenden Eigenschaften unterscheiden ein relationales Datenbanksystem von einem nicht relationalen (wobei hier die der relationalen Datenbank angegeben werden)

- •) keine Zugriffsabhaengigkeiten - eine Relation kann existieren **ohne** die vorherige Definition von Relationen.
- •) keine Abhaengigkeit von Tupeln.
- •) keine Abhaengigkeit fuer Datenbankzugriffe
- •) keine Insert, Update oder Loeschabhaengigkeiten (z.B. Child geloescht, wenn Parent geloescht)
- •) high-level, non-prozedurale set-orientiere Qualifikationsgrundlagen
- •) die Moeglichkeit der Auswahl eines Windows fuer einen Benutzer aus der Menge der Relationen.
- •) die Moeglichkeit semantischer Integritaet waehrend des Laufzeit der Applikation.

2.5.1.5 INTERFACES

Das Kapitel ueber die Interfaces enthaelt: Abfragesprache, Datenbankschemadefinition, Datenbankaenderung, zwingende Definitionen, Datenbankgenerierung, Datenbank= neudefinierung und -neubennenung, Berichterstattung, Dateneingang, Sicherheit, Datenbankkontrolle, Indizes, Speicherplatzdefinition, Zugriffspfade, Datenbankdesign ...

2.5.1.6 DOKUMENTATION

Liste der Dokumentation, die fuer dieses System zugaenglich ist.

2.5.2 DATENBANKRICHTLINIEN

Festlegung der speziellen Datenbankrichtlinien.

2.5.2.1 GENERELLE BESCHREIBUNG

enthaelt die Umschluesselung der Begriffe wie Datenbank, Relation,View, Tuple, Attribut, Domain auf die in der Datenbank entsprechenden Bezeichnungen.

2.5.2.2 DATENBANK

Eine aeusserst grobe Beschreibung der Datenbank:

DATENBANKSTRUKTUR

Beschreibung der Datenbankstruktur auf dem Schema (Typ-) level und auf dem Wertlevel. Beschreibung des Mechanismus der Identifikation einer Datenbank.

DATENBANKOPERATIONEN

Auflisting der Operationen fuer die Definition, Generierung und Manipulierung einer Datenbankstruktur.

2.5.2.3 RELATION

STRUKTUR DER RELATION

Darstellung der Systemdefinition der Relation: wie ist die dominierende Darstellung der Relation (Tabelle von Zeilen und Spalten); Beschreibung der Identifikation der Relation; Beantwortung der folgenden Fragen:
Wie ist die Rolle der Tupel und Attribute in der Relationstruktur? Sind doppelte Tupel erlaubt? Koennen doppelte Namen vergeben werden?

OPERATIONEN DER RELATIONEN

Auflistung der Operationen fuer die Definition und die Manipulierung der Relation.

2.5.2.4 VIEWS

STRUKTUR

Angabe der Systemdefinition eines Views. Wie unterscheidet sich die Viewstruktur von der Relationstruktur? Beschreibung der Namensgebung der Views.

OPERATION

Auflistung der Operationen fuer die Definition und Manipulierung der Views.

2.5.2.5 TUPEL

STRUKTUR

Angabe der Systemdefinition fuer ein Tupel. Beschreibung einer Tupelstruktur (z.B. Recordtype) und Tupelwert. Die Rolle der Attribute und Domaene in der Tupelstruktur. Sind Schluessel definierbar? Beschreibung des Mechanismus fuer eindeutige Schluessel.

OPERATIONEN

Liste der Operationen fuer die Definition und Manipulierung von Tupel. Sind es explizit oder implizit tupel-orientierte Operationen (READ,WRITE, REPLACE) oder Kontrollstruktur.

2.5.2.6 ATTRIBUTE

STRUKTUR

Angabe der Systemdefinition eines Attributes.
Auflistung der Operationen fuer die Definition und Manipulierung der Attribute (z.B. Vergleichsoperationen wie <,>, ...; arithmetische Operatione +,-.....; Funktionen wie Sum,..... usw).
Angabe der Kompatibiltaet der Regeln

2.5.2.7 DOMAINE

STRUKTUR

Angabe der Systemdefinition der Domaine. Ist ein Domain ein Set bestehend aus fundamentalen Objekten oder Entities, ein Set von Werten, ein Teil der Typbeschreibung. Welche Teile der Domaine sind vordefiniert oder welche koennen vom Benutzer selbst definiert werden?

OPERATIONEN

Liste der Operationen (Vergleiche, arithmetische oder vergleichende Operationen). Wie ist die Kompatibiltaet fuer diese Operationen? Koennen Domainoperationen definiert werden oder neudefiniert?

GRENZEN

z.B. Ober und Untegrenzen usw.

2.5.3 FUNKTIONALE FAEHIGKEITEN

Die funktionalen oder operationale Faehigkeiten des System oder der Sprachen werden hier beschrieben.

2.5.3.1 QUALIFIKATION:

Dieses Kapitel befasst sich mit der Philosophie die fuer die Selektion gilt.
Die generelle Beschreibung der Qualifikation beinhaltet:

- .) die Darstellung des Selektionsmechanismuses (algbra- oder kalkuelorientiert), die Moeglichkeiten und Reichweiten der Relation.
- .) die Interfaces

RESTRIKTIONEN

Auflistung und Definition der einfachen Konditionen (z.B. <,<=,...)

SET OPERATIONEN

Auflistung der Operatoren wie z.B. Union, Differenz und erweitertes kartesisches Produkt

JOINING (VERBINDUNG)

Auflistung und Definition der Verbindungsoperatoren z.B. natuerlicher Join, Equi-Join. Wieviele Relationen koennen miteinander verbunden werden? Koennen Relationen mit sich selbst verbunden werden?

2.5.3.2 AENDERUNGEN

Das Ergebnis einer Datenbankaenderungsoperation ist ein neuer Status.

INSERT

Welche Datenbankbestandteile koennen mit Hilfe des Insertkommandos (Relationen, Views, Tupels, Attribute oder Domains) geaendert werden. Charakterisierung der Definition der Insertoperation.

Die Rolle der Tupel, Attribute, Domains bei der Insertoperation.

Koennen die Resultate bei den Abfragen als Parameter fuer Insertoperationen verwendet werden? Angabe eines Beispieles!!

DELETE

Welche Datenbankbestandteile koennen mit Hilfe des Deletekommandos (Relationen, Views, Tupels, Attribute oder Domains) geloescht werden? Charakterisierung der Definition der Deleteoperation.

Die Rolle der Tupel, Attribute, Domains bei der Deleteoperation.

Koennen die Resultate bei den Abfragen als Parameter fuer Deleteoperationen verwendet werden? Angabe eines Beispieles!!

MODIFY

Welche Datenbankbestandteile koennen mit Hilfe des Modifykommandos (Relationen, Views, Tupels, Attribute oder Domains) geaendert werden? Charakterisierung der Definition der Modifyoperation.

Die Rolle der Tupel, Attribute, Domains bei der Modifyoperation.

Koennen die Resultate bei den Abfragen als Parameter fuer

Modifyoperationen verwendet werden? Angabe eines Beispieles!!

COMMIT

Welche Datenbankbestandteile koennen mit Hilfe des Commitkommandos (Relationen, Views, Tupels, Attribute oder Domains) geaendert werden? Charakterisierung der Definition der Commitoperation.

Die Rolle der Tupel, Attribute, Domains bei der Commitoperation.

Koennen die Resultate bei den Abfragen als Parameter fuer Commitoperationen verwendet werden? Angabe eines Beispieles!

WEITERE AENDERUNGSMOEGLICHKEITEN

Beschreibung weiterer Aenderungsmoeglichkeiten.

GRENZEN

Beschreibung der Systemgrenzen.

ARITHMETISCHE UND STTRINGOPERATIONEN

Beschreibung der arithmetischen und Stringoperationen.

SORTIERUNG

Beschreibung des oder der Sortieralgorithmen.

BIBLIOTHEKSFUNKTIONEN

Auflisting der Bibliotheksfunktionen.

FUNKTIONEN - VOM BENUTZER DEFINIERT

Beschreibung des Mechanismus, wie sich ein Benutzer selbst Funktionen definieren kann.

MULTI-TUPEL AENDERUNGEN

Kann mehr als ein Tupel von einer Relation geaendert werden (z.B. eingefuegt, geaendert,...)

GROUPING

Beschreibung der Moeglichkeiten fuer grouping-Tupels. (z.B. partitioning Sets von Tupels)

2.5.4 DEFINITIONEN, GENERIERUNG und ADMINISTRATIONSMOEGLICHKEITEN

2.5.4.1 DEFINITIONSMOEGLICHKEITEN

Die folgenden Kapitel beschaeftigen sich mit der Definition der Datenbankbestandteile, die dann ein Datenbankschema formen.

BESCHREIBUNG DER DB-DEFINITION

Angabe der Liste der Moeglichkeiten einer Datenbankschema-definition

DEFINITION WEITERE DATENBANKMOEGLICHKEITEN

z.B. Prozeduren, Snapshots

MOEGLICHKEITEN MEHRERER DB-VERSIONEN

Beschreibung der Definition von mehreren Datenbankversionen.

2.5.4.2 DATENBANKWIEDERDEFINITION

Waehrend der Lebensdauer einer Datenbank kann es notwendig erscheinen eine Datenbankdefinition zu aendern, das kann so einfach sein, wie Neubenennung oder erfordert den gesamten Initialisierungsvorgang noch einmal durchzufuehren.

NEUBENNENUNG DER DATENBANK

Beschreibung der Neubennenung von Tabellen, Feldern etc.

DEFINIERUNG DER DATENBANK

Beschreibung des Mechanismus bei der Wiederdefinition der Datenbank.

DATENBANKREORGANISATION

Beschreibung des Reorganisierens der Datenbank.

SYSTEMKONTROLLE

Beschreibung der Kontrolle durch das System.

DATENBANK-DICTIONARY

Wo befindet sich derselbe und wie kann man ihn erstellen.

2.5.5 INTERFACES UND DBMS-ARCHITEKTUR

2.5.5.1 SYSTEMARCHITEKTUR

Beschreibung der Systemarchtektur.

2.5.5.2 INTERFACE-BESCHREIBUNG

Beschreibung der Schnittstellen zum System etc.

2.5.6 OPERATIONALE ASPEKTE

2.5.6.1 SICHERHEIT

ZUTRITTSBERECHTIGUNGEN

Beschreibung der Zutrittsberechtigungen, wer hat welche Privilegien?

FAEHIGKEIT

Beschreibung der Faehigkeit des System in Bezug auf die Datensicherheit.

2.5.6.2 PHISIKAL. INTEGRITAET

CRASH-RECOVERY

Welche Aktionen werden bei einem Systemzusammenbruch getaetigt?

2.5.7 ENVIRONMENT

SOFTWARE

Beschreibung der Systemsoftware.

HARDWARE

Beschreibung der Systemhardware.

3 ALLGEMEINES UEBER DIE IMPLEMENTIERUNG

3.1 BESCHREIBUNG DER DATENBANK NETZ

3.1.1 ALLGEMEINES

3.1.1.1 BESCHREIBUNG

Die Einfuehrung von rechnergesteuerten Fernsprechvermittlungsstellen mit digitaler Sprechwegdurchschaltung in bestehenden Fernsprechnetzen erfordert umfangreiche Untersuchungen hinsichtlich technischer Durchfuehrbarkeit und wirtschaftlich optimaler Realisierung.

Grundlage fuer die Erarbeitung von Einfuehrungsstrategien und die Planung von Teilprojekten ist die moeglichst vollstaendige Erfassung der Struktur des jeweiligen Fernsprechnetzes speziell in Bezug auf Groesse und Standort der Vermittlungsstellen.

Das Datenbanksystem "NETZ" wurde zur Erfassung der Netzstruktur auf der Basis von REXX und GERM entwickelt, mit dessen Hilfe die Daten gespeichert, verwaltet und leicht auf aktuellem Stand gehalten werden koennen.

3.1.1.2 AUFRUF

Es gibt fuer die Datenbank Netz 2 Arten von Permission, naemlich die read- oder write-Permission. Der Benutzer kann sich aussuchen, ob er von der Datenbank nur lesen will oder auch schreiben will.

Unter Schreiben ist prinzipiell zu verstehen:

- .) LADEN der Datenbank
- .) UPDATE der Datenbank
- .) LOESCHEN der Datenbank

Unter Lesen versteht man im Prinzip:

- .) DRUCKEN der Datenbank
  - ..) in lesbarer Form
  - ..) wie es intern im Germ abgespeichert ist
- .) AUSWERTEN der Datenbank

Mit dem Aufruf NETZ R erhaelt man die READ-Permission, mit dem Aufruf NETZ W die WRITE-Permission, wobei innerhalb des Programmes noch die Userid ueberprueft wird..

ist die Userid ungleich einer Benutzerkennung, die auf einem File steht, erhaelt man nur READ-Permission.

Je nach der Permission, die nun dem Benutzer zugestanden wird, wird die "DATENBANKPLATTE" im entsprechenden Mode (R/W) gelinkt.

3.1.1.3 BEMERKUNGEN

Da die Datenbank Netz eine relationale Datenbank ist, sind alle Daten in Tabellen gespeichert;
neben den sogenannten USER ENTITY SET (= Tabellen, die die Daten des Benutzers enthalten)
gibt es noch die SYSTEM ENTITY SETS.

Die SYSTEM ENTITY SETS werden automatisch zum Zeitpunkt des Entstehens der Datenbank (mittels des Command CREATE) erzeugt. Sie enthalten die Namen
und die Beschreibung der USER ENTITY SETS und die Verbindungen (Relationen) zwischen den einzelnen USER ENTITY SETS.

3.1.2 Erste Ausbaustufe

In der 1. Ausbaustufe handelt es sich um eine nonsecure Datenbank mit read- und write-password.
Der Nachteil einer nonsecuren Datenbank besteht darin, dass jeder der das read und write-password kennt uneingeschraenkten Zugriff auf die Datenbank hat.

3.1.3 Zweite Ausbaustufe

In der 2. Ausbaustufe handelt es sich dann um eine secure Datenbank. Damit wird jedem Benutzer ein WINDOW zur Verfuegung gestellt - und es kann dann genau definiert werden, welcher Benutzer welche Zugriffsrechte hat und ob er nur Lesen, Schreiben, Loeschen etc... in seinem Bereich darf.
Jedes Window ist zusaetzlich mit einem Password geschuetzt.

3.1.4 Dritte Ausbaustufe

Hier ist vorgesehen, dass mehrere "Netz-datenbanken" parallel existieren koennen und zwar mit der Kennung des Jahres: Netz83, Netz84, ...

Ausserdem ist vorgesehen, dass vergleichende Auswertungen

zwischen den dann parallel gehaltenen Datenbanken moeglich sind.

3.1.5 aktueller Stand

Momentan koennen mehrere "Netz-datenbanken" parallel existieren (d.h. die dritte Ausbaustufe wurde realisiert); ausserdem gibt es zusaetzlich zu den Auswerteprogrammen noch eine Online-moeglichkeit fuer den Benutzer.

Die Onlinemoeglichkeit gibt den Benutzer die Moeglichkeit sich die einzelnen Tabellen (abhaengig vom Jahr) mit einfachen Germ-Befehle anzuschauen, zu drucken und einfache Auswertungen zu taetigen. Ein entsprechendes Helpfacility wurde ebenfalls geschaffen.

3.2 ALLGEMEINE BESCHREIBUNG DER TABELLENINHALTE

Intern werden alle Sonderzeichen und die Blanks innerhalb eines Feldelementes durch "$" ersetzt; wenn sich der Benutzer die Tabelle, wie sie im Germ gespeichert ist, anschaut, so muss er sich alle $ durch Blanks ersetzt denken. Aus diesem Grund gibt es zwei Arten von Druckprogrammen

- drucken der Tabellen, wie sie im Germ gespeichert sind; hier erscheint als Trennzeichen statt des sonst ueblichen Blanks das "$"

- drucken in "menschlich" lesbarer Form; alle "$" sind durch Blanks ersetzt.

Die fuer diesen Vorgang notwendigen Unterprogramme lauten KA und KAR.
Die erstere wandelt die Blanks und alle erlaubten Sonderzeichen (in allen Inpufiles und bei allen Onlineeingaben sind ausser A .. Z (Gross- und Kleinbuchstaben), 0 .. 9, Blank, "/", ",", ".", "(", ")" und "-" keine weiteren Sonderzeichen erlaubt - die oben angefuehrten Sonderzeichen werden durch "$" (sind alle ausser A .. Z und 0 .. 9) ersetzt), die letztere ist fuer den umgekehrten Vorgang von Bedeutung - sie wandelt den Delimiter "$" wieder in die Blanks um.

3.3 KENNZEICHENKATALOG FUER GERM

3.3.1 EINFUEHRUNG

Im nachfolgenden Kapitel wird Germ anhand des Kennzeichenkataloges analysiert; wobei der Fettdruck spezielle Informationen ueber die Implementierung enthaelt.

3.3.1.1 IDENTIFIKATION:

GERM (General Entity Relationship Model) ist von BNR (Bell Northern Research in Kanada) entwickelt worden.

3.3.1.2 STATUS

SYSTEM

Momentan wird vom Benutzer die 11.1-te Version verwendet.

APPLIKATIONEN

GERM wurde von der BNR entwickelt und dient der raschen Wiedererhaltung der Information aus einer Datenbank. Die Anwendung liegt weniger im kommerziellen als im technischen Bereich und da vor allem im Entwicklungsbereich.

Auch das neu entwickelte Datenbanksystem "NETZ" wurde mit Hilfe von GERM entwickelt. Der Zweck dieses System besteht vor allem darin, dass man, wie es bei jedem Datenbanksystem moeglich ist, die Daten loeschen, laden und aendern kann; ein weiterer Zweck besteht darin, dass man statistische Auswertungen laufen lassen kann, wie Netzaufstellung, Groessenbestimmung und Ersetzung der bisherigen Aemter durch zukuenftige digitale Aemter.

Der Benutzer hat auch noch die Moeglichkeit, ONLINE die Integritaetsbeziehungen fuer die Daten anzuschauen, zu aendern und einzugeben.

3.3.1.3 SYSTEMHINTERGRUND

GERM ist ein Programmpaket, das unter VM laeuft. Virtual Machine Facility/370 oder auch VM/370 ist der Name fuer eine Menge von Programmen, die die IBM-Maschine

3270 ueberwachen.

Die Environments sind aus folgenden Gruenden wichtig:

- .) Verschiedene Tasks benoetigen verschiedene Environments.
- .) Zwischen verschiedene Environments muss man sich natuerlich beliebig zwischen ihnen bewegen koenne.

BASIC VM ENVIRONMENT - CP AND CMS

CONTROL PROGRAM (CP)

CP kontrolliert und ueberwacht die physische Maschine. Auch jedes einzelne Terminal wird durch das CP ueberwacht. CP ist ein sehr maechtiges Programm, das einer Menge von Benutzern erlaubt den Computer gleichzeitig zu verwenden. CP sorgt dafuer, dass es die Anweisungen und Befehle, die ein Benutzer ausfuehrt keinen weiteren Benutzer behindert oder stoert. CP erlaubt dem Anwender KEINE Fehler, die der Maschine "ernsthaft schaden" koennte.

CONVERSATIONAL MONITOR SYSTEM (CMS)

CMS-Commands haben Auswirkungen auf die Files des Benutzers.

WECHSEL D. ENVIRONMENTES

Wenn man "einloggt", so ist dies ein typisches Beispiel fuer den Wechsel des Environments. Der CP-Environment ist automatisch zur LOGON-Zeit vorhanden. Wenn nun das CP-Logon komplett ist, so muss der Benutzer die Entertaste betaetigen und wechselt nun vom CP-Mode zum CMS-Mode.

Wenn man nun GERM verwendet, so muss man sich zusaetzlich noch in den GERM-Mode begeben; dies geschieht in 2 Schritten:

- .) Eingabe von GERMDISK fuehrt zum Linken der notwendigen Platten.
- .) Eingabe von GERM bewirkt, dass sich der Benutzer im GERM-Mode befindet.

Bei dem neuentwickelten Datenbanksystem "NETZ" wird der Wechsel des Environmentes CMS - GERM von den Programmen selbst getaetigt:

- **.) Der Aufruf GERMDISK, der dem Linken der notwendigen Systemplatten dient, wird einmal getaetigt und zwar beim Aufruf des Programmpaketes mit Hilfe von**

NETZ R oder NETZ W durchgefuehrt.

•) die Eingabe von GERM, die dem direkten Einstieg in GERM-Mode entspricht, wird bei jedem Datenbankzugriff getaetigt z.B. beim Listen, Laden .....

Der Aufruf wird folgendermassen getaetigt:

Germmacroname + Parameter werden gespeichert. (wobei auch mehrere Macros gespeichert werden koennen Befehl BYE (Ausstieg vom Germ)

Durchfuehrung des Befehles GERM bewirkt den Einstieg in das Germ; und dabei werden dann der Reihe nach die gespeicherten Befehle durchgefuehrt.

3.3.1.4 NOTWENDIGE RELATIONALE CHARAKTERISTIKEN

In GERM sind Relationen in Form von zweidimensionalen Tabellen repraesentiert.
Alle Daten in GERM sind in Tabellen gespeichert, sogenannte ENTITY-SETS. Ein ENTITY-SET ist eine Ansammlung von Records (auch Entities genannt) unter einem einzigen Namen. In GERM sieht jeder Entitiy im wesentlichen wie eine Tabelle.
Jede Spalte ist ein Attribut (auch Feld genannt).

Eine GERM-Datenbank besteht aus 2 Arten von Entity Sets:

•) USER ENTITY SETS: Diese Tabellen enthalten die Daten, die vom Benutzer eingegeben wurden.

•) SYSTEM ENTITY SETS: diese Tabellen entstehen automatisch bei der "Erschaffung" der Datenbank. In diesen Tabellen sind die Namen und Beschreibungen der User entity Sets und die Relationsship zwischen den Tabellen enthalten.

3.3.1.5 DOKUMENTATION

Die Dokumentation enthaelt folgende Manuals:

- GERMIM ..... Germ Einfuehrungsmanual
- GERMRM ..... Germ Refernece Manual
- GERMFD ..... Germ Interactive Form Definition Facility
- GERMDC ..... Guide to setting up Databases
- GERMGI ..... Discusses the use of GERM Index files
- GERMFI ..... Files, definitiones and sorting

3.3.1.6 GENERELLE SYSTEMBESCHREIBUNG

3.3.2 DATENBANKRICHTLINIEN

3.3.2.1 GENERELLE BESCHREIBUNG

Kennzeichenkatalog= bezeichnung	GERM-Bezeichnung
Datenbank	Datenbank
Relation	Tabelle
View	Window
Tuple	Entity, Zeile
Attribut	Spalte
Domain	Domain

3.3.2.2 DATENBANK

DATENBANKSTRUKTUR

Im GERM gibt es zwei Arten von Relationship.

- CONTENT RELATIONSHIP
  verbindet die "Zeilen" einer Tabelle mit denen einer anderen Tabelle abhaengig von einem Datenelement, das in beiden Tabellen vorkommt.

- CONNECT RELATIONSHIP
  verbindet die "Zeilen" einer Tabelle mit denen einer anderen Tabelle, wie es der Benutzer definiert, wobei nichts an den Daten die Verbindung ahenen laesst.

DATENBANKOPERATIONEN

Im GERM sind folgende Befehle von Bedeutung:

- CREATE datenbankname
  Kreiert eine neue Datenbank

- OPEN datenbankname (R/W)
  eroeffnet die Datenbank

- ADD FIELDS
  Fuegt Feldelemente hinzu

- FIND tabellenname WHERE expression
  sucht jene Elemente aus der angegebenen Tabelle, fuer die die Expression zutrifft.

- LIST all(oder Anzahl)
  listet die gefundenen Elemente auf

- CHANGE
  aendert die Daten

ZWINGENDE DATENBANKREGELN

System Entity Sets werden vom System angelegt .
Die Systemfiles lauten:
Filename=Datenbankname
Filetype=GERM(E fuer Entity I fuer Index)###
... 3-stellige Nummer > 900

Filemode=a,.....,z (je nachdem was im Profile angegeben wurde -
Default = a)
Der Filemode ist eine Adresse fuer eine Minidisk.

Man kann im Germ 2 Arten von Datenbanken kreieren

- die nonsecure Datenbank
- die secure Datenbank

Bei einer nonsecure Datenbank kann man (muss man jedoch nicht) ein Read- und/oder Write-password angeben. Wenn jedoch jemand diese Passwoerter kennt, hat er uneingeschraenkten Zugriff zu den Daten.

Bei einer secure Datenbank kann nur der Datenbankadministrator

DATENBANKOPERATIONEN

Im GERM sind folgende Befehle von Bedeutung:

- CREATE datenbankname
  Kreiert eine neue Datenbank

- OPEN datenbankname (R/W)
  eroeffnet die Datenbank

- ADD FIELDS
  Fuegt Feldelemente hinzu

- FIND tabellenname WHERE expression
  sucht jene Elemente aus der angegebenen Tabelle, fuer die die Expression zutrifft.

- LIST all(oder Anzahl)
  listet die gefundenen Elemente auf

- CHANGE
  aendert die Daten

ZWINGENDE DATENBANKREGELN

System Entity Sets werden vom System angelegt .
Die Systemfiles lauten:
Filename=Datenbankname
Filetype=GERM(E fuer Entity I fuer Index)###

... 3-stellige Nummer > 900

Filemode=a,.....,z (je nachdem was im Profile angegeben wurde - Default = a)

Der Filemode ist eine Adresse fuer eine Minidisk.

Man kann im Germ 2 Arten von Datenbanken kreieren

- die nonsecure Datenbank
- die secure Datenbank

Bei einer nonsecure Datenbank kann man (muss man jedoch nicht) ein Read- und/oder Write-password angeben. Wenn jedoch jemand diese Passwoerter kennt, hat er uneingeschraenkten Zugriff zu den Daten.

Bei einer secure Datenbank kann nur der Datenbankadministrator

(mit einem Masterpassword) die Benutzersichten definieren und die Permission an die Benutzer vergeben.

3.3.2.3 RELATION

STRUKTUR DER RELATION

Jede Datenbank repraesentiert einen Aspekt der realen Welt und kann mit Hilfe **einer** Tabelle d.h.mit Hilfe eines Entity Set dargestellt werden.
Diese Art der Repraesentation fuehrt zu einer sehr hohen Datenredundanz.

Eine Relation, in GERM Entity genannt, ist im wesentlichen ein Set von n Tupeln dargestellt in einer 2-dimensionalen Tabelle.

Auch in dem Datenbanksystem "NETZ" sind die Daten auf mehrere Tabellen aufgeteilt, um auf diese Weise Redundanz zu vermeiden und auch den Speicherplatz moeglichst gering zu halten.

**So sind fuer Bezeichnungen zwei extra Tabellen und fuer die Technik eine weitere Tabelle erstellt worden. Diese Tabellen enthalten im Prinzip nur die Abkuerzungen und die Bezeichnung voll ausgeschrieben.
Wenn man in den anderen Tabellen so eine Bezeichnung gerne voll ausgeschrieben haette und das direkt ohne weitere Tabellen abgespeichert haette, so wuerde dies zu einer zu grossen Redundanz und zu einem zu grossen Speicherplatzverbrauch fuehren.**

Will der Benutzer einen Ausdruck der Technik oder der Bezeichnung auch ohne eine direkte Speicherung voll ausgeschrieben haben, so kann er sich dies leicht mit Hilfe einer Relation schaffen.

Wie bereits erwaehnt gibt es beim Datenbanksystem "NETZ" mehrere Ausbaustufen bezueglich der Sicherheit. Bei der nonsecuren Datenbank wird die "Datenbankplatte" noch durch einen zusaetzlichen Effekt geschuetzt:

> **Der Benutzer gibt an, ob er Read- oder Writepermission haben will. Ist die Userid ungleich der des Datenbankadministrators, so erhaelt der Benutzer automatisch die Read-Permission.
> Aufgrund dieser Permission wird dann die Platte im entsprechenden Mode gelinkt und ist somit**

entsprechend durch das CMS geschuetzt.

3.3.2.4 VIEWS

STRUKTUR DES VIEWS

Bei einer secure Datenbank kann man zusaetzlich noch jedem Benutzer ein sogenanntes Window zuordnen, indem man diesen auf die Tabellen, die fuer ihn von Bedeutung sind, zuzugreifen laesst. (entweder im Read- oder Write-modus).

D.h. das Security-system des GERM definiert die Privilegien fuer jedes Window und diese Windows werden allein vom Datenbankadministrator verwaltet.

OPERATION DER VIEW

Die Windows werden vom Datenbankadministrator vergeben und definiert, somit auch von diesem verwaltet.

Der Benutzer kann sein Window mit folgendem Kommando aktivieren:

```
USE VIEW view_name <View_password>
```

Man kann dem Benutzer folgende Privilegien zuordnen:

```
READ
ADD
CHANGE
DELETE
```

wobei in austeigender Reihenfolge der Privilegien definiert sind; wer den DELETE-Zugriff hat, hat also auch CHANGE,ADD und READ-Zugriff.

GRENZEN DES VIEWS

Mit Hilfe des VIEW hat jeder Benutzer nur eine Sicht auf die Datenbank, d.h. ihm steht (ausser in seltenen Ausnahmen) nie die ganze Datenbank zur Verfuegung.
D.h. dem Benutzer wird somit nur eine bestimmte logische Sicht der Datenbank zugeordnet.

Jedes Window kann zusaetzlich noch mit einem Passwort geschuetzt sein.

3.3.2.5 TUPEL

STRUKTUR DER TUPEL

Ein Tupel ist eine Zeile in einer Tabelle. Es besteht aus Attributen, die ueber die Domains definiert sind. Eine Zeile ist implizit definiert, wenn die Tabelle definiert wird und zwar durch die Laenge, Domain, Key (d.h. unique) usw.

Die Schluessel werden bei der Definition der Relation angegeben und zwar beim ADD FIELDS.

OPERATIONEN VON TUPEL

Tupel koennen eingefuegt, geloescht, geaendert und gelistet werden.

Die Tupel sind nicht sortiert. Eine bestimmte Sortierreihenfolge kann gedoch gewuenscht werden.

Bei einem Update einer Tabelle und speziell beim nachtraeglich Einfuegen eines Elementes, wird dieses nicht an einer speziell gewuenschten Stelle eingefuegt, sondern am Ende der Tabelle.

Auf diese Weise erspart man sich:

- **die Pointerverbindungen, wenn man eine spezielle Reihenfolge aufrechterhalten moechte.**
- **oder das Neusortieren vor dem endgueltigen Abspeichern.**

3.3.2.6 ATTRIBUTE

STRUKTUR DER ATTRIBUTE

Ein Attribut ist der Name (die Ueberschrift) einer Spalte. Die Attribute werden abhaengig von den Domains definiert.

Die Definition einer Tabelle innerhalb einer GERM-Datenbank geschieht in 2 Schritten:

- ADD ENTS:
  legt den Namens der Tabelle innerhalb einer Datenbank fest
- ADD FIELDS:

legt die Attribute der Tabelle fest:
es werden pro Attribut folgende Informationen verlangt:

- Feldname
- Besitzer = tabellenname
- Index yes .... es wird ein Index generiert.
  no ..... kein Index
- DUPS = no, wenn index=yes (moeglich)
  - entspricht unique
- Type char oder Number

OPERATIONEN DER ATTRIBUTE

Es gibt folgende Operatoren:

unaeres +,
unaeres -,
folgende arithmetische Operatoren: +,-,/,*,

Vergleichsoperatoren: =,>,<,¬=,<=,>=,
boolsche Operatoren: AND,OR,NOT

vordefinierte Funktionen:

SUBSTRing (string_expression n m)
erstellt einen Teilstring von der Stelle n
m Stellen lang

CONCATenate (string_Expression1 string_expression2 n)
verbindet die zwei Stringexpression mit
n Blanks

SLENGTH (string_expression)
stellt die Laenge der Stringexpression fest

CHOP (string_expression)
entfernt alle Blanks

TRANSLATE (String_expression1 string_expression2
string_expression3 n)
ersetzt n-mal in string_expression1
string_expression2 durch string_expression3

3.3.2.7 DOMAINE

STRUKTUR DER DOMAINE

Domain Datentypen sind:

CHARAKTER	n	
NUMBER	n	m

PACKED	n	m
INT	n	

n ..... (Vorkomma-) Stellenanzahl
m ..... Nachkommastellenanzahl

3.3.3 FUNKTIONALE FAEHIGKEITEN

3.3.3.1 FUNKTIONEN

INSERT

Mittels Insertkommandos koennen folgende Datenbankbestandteile (Relationen, Views, Tupels, Attribute oder Daomains) geaendert werden.

DELETE

Relationen, Views, Tupels, Attribute koennen mit Hilfe des DELETE-Kommandos geloescht werden. Zuerst muss das gesuchte Element oder die gesuchten Elemente mittels FIND-Kommandos gefunden werden und dannach koennen die neu gefundenen Elemente geloescht werden.

MODIFY

Relationen, Views, Tupels, Attribute koennen mit Hilfe des CHANGE-Kommandos geaendert werden. Zuerst muss das gesuchte Element oder die gesuchten Elemente mittels FIND-Kommandos gefunden werden und dannach koennen die neu gefundenen Elemente geaendert werden.

SORTIEREN

GERM hat 2 exzellente Sortierroutinen. Trotzdem stellt das Sortieren von grossen Tabellen einen grosser Aufwand dar. Die richtige Auswahl von den Sortroutinen kann eine Einsrung der CPU-Zeit bringen.

Der erste Befehl **SORT** ist optimal
in der CPU-Zeit fuer kleinere Tabellen (<= 6500 Records)
Ein Record braucht weniger als 0.04 CPU-Sekunden.

Der zweite Befehl **OSSORT** hat gute CPU-
Zeiten fuer grosse Tabellen (> 6500 Records)
0.8 CPU-Sekunden fuer ein Record

BIBLIOTHEKSFUNKTIONEN

Da sich der Benutzer selber Funktionen und Batch-Programme in Form von GERMMACROS schreiben kann , koennen diese in Bibliotheken vorhanden sein.

FUNKTIONEN - VOM BENUTZER DEFINIERT

Der Benutzer kann sich Funktionen in Form von Germmacros definieren

DEFINITION WEITERE DATENBANKMOEGLICHKEITEN

z.B. Prozeduren=Germmacros

MOEGLICHKEITEN MEHRERER DB-VERSIONEN

Im Germ gibt es die Moeglichkeit mehrere Datenbankversionen geladen zu haben (unter verschiedenen Namen), aber Querverweise zwischen den Datenbanken zwecks Auswertung sind nicht moeglich, da in dem Moment, wo man eine Datenbank eroeffnet, die letztoffene geschlossen wird.

3.3.3.2 DATENBANKWIEDERDEFINITION

Waehrend der Lebensdauer einer Datenbank kann es notwendig erscheinen, dass man eine Datenbankdefinition veraendern muss; dies geschieht indem man ein neues Create der Datenbank macht und alle Tabellen neu definiert (aendert sich die Struktur nicht, so ist kein Neuladen notwendig sondern nur Tabellenaenderung und dann UPDATE ENT DEF ALL oder UPDATE ENT DEF tabellenname).

DATENBANKBESTANDTEILENEUBENNUNG

Mittels ADD ENTS und ADD FIELDS und danach UPD ENT DEF ALL oder UPD ENT DEF tabellenname

DATENBANKBESTANDTEILENEUDEFINIERUNG

Mittels ADD ENTS und ADD FIELDS und danach UPD ENT DEF ALL oder UPD ENT DEF tabellenname

DATENBANKREORGANISATION

Waehrend der Lebensdauer einer Datenbank kann es notwendig erscheinen, dass man eine Datenbankdefinition veraendern muss; dies geschieht indem man ein neues Create der Datenbank macht und alle Tabellen neu definiert (aendert sich die Struktur nicht, so ist kein Neuladen notwendig sondern nur Tabellenaenderung und dann UPDATE ENT DEF ALL oder UPDATE ENT DEF tabellenname).

SYSTEMKONTROLLE

Da die Datenbanktabellen (SYSTEM ENTITY SET und USER ENTITY SET) auf der Platte abgespeichert werden, unterliegen sie der File- und Systemkontrolle vom CMS.

3.3.4 OPERATIONALE ASPEKTE

3.3.4.1 SICHERHEIT

nonsecure und secure Datenbank bereits vorher erklaert.

ZUTRITTSBERECHTIGUNGEN

In Germ gibt es wie bereits erwaehnt 2 Arten von Datenbanken die nonsecure und die secure Datenbank.

3.4 VORTEILE VOM GERM

3.4.1 EINFACHHEIT

Die Datenobjekte sind nur Relationen und diese basieren nur auf ein paar Operationen der relationalen Datenbank; auch die Abfragesprache beruht auf den relationalen Kalkuelen.

3.4.2 EINHEITLICHKEIT

Eine einzige Sprache behandelt die Datendefinition, Datenbankaenderungen, Listungen und Zugriffsberechtigungen-festlegung.

3.4.3 SICHERHEIT

Das Sicherheitsssystem ist relativ gut ausgelegt und gut ausgefeilt.

3.4.4 DATENUNABHAENIGKEIT

Der Benutzer hat volle Datenunabhaengigkeit; das einzige Problem welches sich stellen kann ist das des Speicherplatzes.

3.5 NACHTEILE VON GERM

Wenn mann grosse Datenmengen hat, so benoetigt das Germ zum Ausfuehren einzelner Befehle relativ lang:

Er benoetigt folgende Cpu-Zeiten fuer die Durchfuehrung folgender Befehle:

- .) Einstieg in die Datenbank, Suchen der Tabelle und Listen von 1783 Elemente:
  4.76 reale CPU-Sec.
  5.59 virtuelle CPU-Sec.

- .) Einstieg in die Datenbank, Suchen der Tabelle und Listen von 13 Elemente:
  0.73 reale CPU-Sec.
  1.08 virtuelle CPU-Sec.

4 ANHANG - LITERATUR

•) Joachim W. Schmidt und Michael Brody :
"RELATIONAL DATABASESYSTEMS Analysis + Comparisons"

•) W. Eggerichs:
"DBASE II - Bd.1 Einfuehrung"

•) Dr. Peter Albrecht"
"Das Datenbanksystem Dbase II"

•) Schlageter/Stucky:
"Datenbanksysteme"

•) BNR - Craig Bishop :
"GERMIM - Das Germ Einfuehrungsmanual"
10. April 1985

•) BNR - Department 8M21 :
"GERMRM - Das Germ Refernece Manual"
19. Oktober 1984

•) BNR - Garth Smedley :
"GERMFD - Das Germ Interactive Form Definition Facility"
22. Oktober 1982

•) BNR - Department 8M21, MER :
"GERMDC - Guide to setting up Databases"
18. Oktober 1984

•) BNR - C.J.M. Turnball :
"GERMGI - Discusses the use of GERM Index files"
July 1980

•) BNR :
"GERMFI - Files, definitiones and sorting"
1981