ENTWURF, REALISIERUNG UND BEWERTUNG

EINER PROGRAMMIERERFREUNDLICHEN DATENBANKVERARBEITUNGSSPRACHE

IM MEDIZINISCHEN SYSTEM HANNOVER

Von der Fakultät für Maschinenbau und Elektrotechnik
der Technischen Universität Carolo-Wilhelmina
zu Braunschweig
zur Erlangung der Würde eines
Doktor-Ingenieurs (Dr.-Ing.)
genehmigte

D I S S E R T A T I O N

von Dipl.-Ing. Werner Weingarten
aus (Geburtsort) Hamburg

eingereicht am: 5. Juli 1980
Mündliche Prüfung am: 5. November 1980
Berichterstatter: Prof. Dr. Stiege
Mitberichterstatter: Prof. Dr. Berr, Prof. Dr. Sauter

1981
(Druckjahr)

G L I E D E R U N G

		Seite
1.	PROBLEMSTELLUNG	
1.1	Umfeld und Thema der Arbeit	
1.1.1	Medizinische Informatik	05
1.1.2	Krankenhausinformationssysteme	
1.1.2.1	Aufgaben eines Krankenhausinformationssystems	05
1.1.2.2	Probleme des Datenverarbeitungsgrundsystems	07
1.1.2.3	Definition einer Sprache als Grundsystemergänzung	07
1.1.2.4	Leistungsfähigkeit der Sprachergänzung	10
1.2	Bezug zur Datenbankforschung	
1.2.1	Datenmodelle	11
1.2.2	Vergleichbare Zielsetzungen	14
1.2.3	Leistungsbewertung von Datenverarbeitungssystemen	14
1.3	Inhalt der Folgekapitel	16
2.	SYNTAXKONZEPT UND SPRACHIMPLEMENTIERUNG	
2.1	Pfad-orientiertes Zugriffs- und Verarbeitungsprinzip	18
2.2	Sprachanweisungen	
2.2.1	Datensatzzugriff	19
2.2.2	Datenelementverarbeitung	19
2.2.3	Kontrollogik	20
2.2.4	Darstellung der Sprachelemente	21
2.3	Sprachimplementierung	
2.3.1	Gliederung in Funktionen	23
2.3.1.1	Hauptprogrammblöcke (Horizontale)	24
2.3.1.2	Unterprogramme (Vertikale)	27
2.3.2	Programmrealisierung	
2.3.2.1	Allgemeines	30
2.3.2.2	Programmablaufgerüst	35
2.3.2.3	Dialog/Stapel-Version	37
2.3.3	Syntaxanalyse und Übersetzer	38
2.3.4	Interpreter	42
2.3.5	Zusätzliche Funktionen im Stapelbetrieb	42

Seite

3.	LEISTUNGSBEWERTUNG	
3.1	Qualitative Bewertung	44
3.2	Quantitative, vergleichende Bewertung	
3.2.1	Methoden der Leistungsermittlung	45
3.2.1.1	Analytisches Verfahren	45
3.2.1.2	Systemsimulation	45
3.2.1.3	Leistungsmessung	46
3.2.1.4	Benchmarktest	46
3.2.1.5	Sog. Schreibtischanalyse	46
3.2.1.6	Schritte zur Leistungsermittlung	46
3.2.2	Motivation für eine vergleichende Leistungsbewertung	47
3.2.3	Suchrechnerkonzept	47
3.2.4	Durchführung der Leistungsermittlung	49
3.2.4.1	Antwortzeitmodelle und ihre Validierung	50
3.2.4.1.1	SAM	53
3.2.4.1.2	DAM	54
3.2.4.1.3	PATSY	55
3.2.4.1.4	HISAM	57
3.2.4.1.5	HDAM	63
3.2.4.1.6	HIDAM	72
3.2.4.1.7	SURE	76
3.2.4.2	Arbeitslasten und Ermittlung der Antwortzeiten	77
3.2.4.2.1	Datenbestände für den Vergleich	
3.2.4.2.2	Trefferhäufigkeit und Komplexität der Anfrage	88
3.2.4.2.3	Patientendatenauskunft, Patientenauswahl	90
3.2.4.2.4	Patientendatenauskunft, Datenoptionen	98
3.2.4.2.5	Auswerteprogramm PATWERT	103
3.2.4.2.6	Patientenentlassungsliste	107
3.2.4.2.7	Akkumulierter Laborbericht	109
3.2.4.2.8	Ad Hoc-Anfragen	110
4.	ZUSAMMENFASSENDE DISKUSSION	123
5.	LITERATURVERZEICHNIS	127
6.	FORMALE SYNTAXDEFINITIONEN	
6.1	Kommandodokumentation	137
6.2	PDPL	144
6.3	SURE	155

1. PROBLEMSTELLUNG

1.1 Umfeld und Thema der Arbeit

1.1.1 Medizinische Informatik

Die Anwendung von Methoden der Informatik in der Medizin gewinnt zunehmend an Bedeutung und manifestiert sich als Medizinische Informatik nicht nur durch eine wachsende Zahl von Systemen und Verfahren, sondern auch durch die Einführung neuer Studiengänge (30, 34, 35, 67). Ganz allgemein sind als Aufgaben die Dokumentation, Analyse, Steuerung, Regelung und Synthese von Informationsprozessen im Gesundheitswesen anzusehen (50, 63, 64). Die zweite Weltkonferenz MEDINFO 1977 in Toronto (85) spiegelte in 23 Sitzungen mit 200 Beiträgen das breit gefächerte Spektrum der Medizinischen Informatik. Schwerpunkte waren Mathematik und Statistik, Biosignalverarbeitung und Informationssysteme. Die hier beschriebene Arbeit entstammt dem letztgenannten Schwerpunkt.

Ein Informationssystem ist ein integriertes System der Datenerfassung, Datenverwaltung, Datenspeicherung, Datenwiedergabe und Steuerung dieses Systems mittels dieser Daten (63, 64). Informationssysteme im Gesundheitswesen sind auf den verschiedenen institutionellen und funktionalen Ebenen wie Weltgesundheitsorganisation, Europäische Gemeinschaft, Bundesrepublik Deutschland, Krankenhaus und Arztpraxis denkbar.

Ein Krankenhausinformationssystem (KIS) (59, 63, 65) ist eine Speziallösung für die Probleme des Großbetriebes Krankenhaus innerhalb des Makrosystems Gesundheitswesen. Informationssysteme sollen helfen, bei verbesserter oder gleicher Qualität der Leistungen, Kosten zu senken. Der allgemeine Ansatz des KIS umfaßt die Administration, die Patientenversorgung, regionale Aufgaben, wie auch für Universitätskliniken die medizinische Forschung.

1.1.2 Krankenhausinformationssystem

1.1.2.1 Aufgaben des Krankenhausinformationssystems

Der Krankenhausbereich in sich ist gekennzeichnet durch komplexe Informationsstrukturen, große Informationsmengen, ein besonderes Schutzbedürfnis seiner Daten (8) und eine Vielfalt von täglich oder in anderen Perioden wiederkehrende Aufgaben (Routineaufgaben) (67, 87, 88). Die Systemstruktur hat darüber hinaus z.B. aufgrund ärztlicher und gesetzgeberischer Aktivitäten ein dynamisches Verhalten. Die administrative Seite ist mit den Informationsmengen befaßt, die dazu dienen, die Patientenaufnahme, -verpflegung, -unterbringung und -abrechnung einerseits und Personal-, Betriebsmittel- und Finanzmanagement andererseits zu ermöglichen.

Die ärztlich medizinische Seite ist mit den Informationsmengen befaßt, die den Patienten, seine Krankheit und deren Behandlung unmittelbar betreffen, wie z.B. die Krankengeschichte mit diagnosti-

schen und therapeutischen Maßnahmen. Die Krankenakte ist hierfür im konventionell geführten Krankenhaus der wichtigste Informationsträger.

Die Hauptproblemursachen für ein Krankenhausinformationssystem (KIS) liegen im großen Informationsvolumen, in der Verteilung der Informationen über verschiedene Disziplinen nach Zeit, Ort, Funktion und anderen Kriterien. Die Kommunikation der Funktionsbereiche ist dadurch erschwert, die ablaufenden Prozesse zeichnen sich durch ungenügende Regelbarkeit und Steuerung aus. Dazu kommt wie bei allen Informationssystemen das Problem der psychologischen Einstellung der Beteiligten zum Computersystem und die hohe Dynamik der Anforderungen an das KIS.
Die Informations- und Steuerungsfunktionen sind nur als integriertes System mit einer den verschiedenen Aufgaben gerecht werdenden Datenbank zu beherrschen (63, 71, 75, 77). Im Medizinischen System Hannover (MSH), dem KIS für die Medizinische Hochschule Hannover, sind administrative und medizinische Daten von 160.000 Patienten mit 210.000 Aufenthalten (1980) in der Datenbank ansprechbar. Unter Datenbank (DB) soll sowohl die Gesamtheit der hierarchisch strukturierten und vom "Information Management System" (IMS) der Firma IBM verwalteten Daten (37), als auch der durch das Betriebssystem OS (38) verwalteten Dateien verstanden werden, soweit sie für Patienten relevante Daten enthalten.

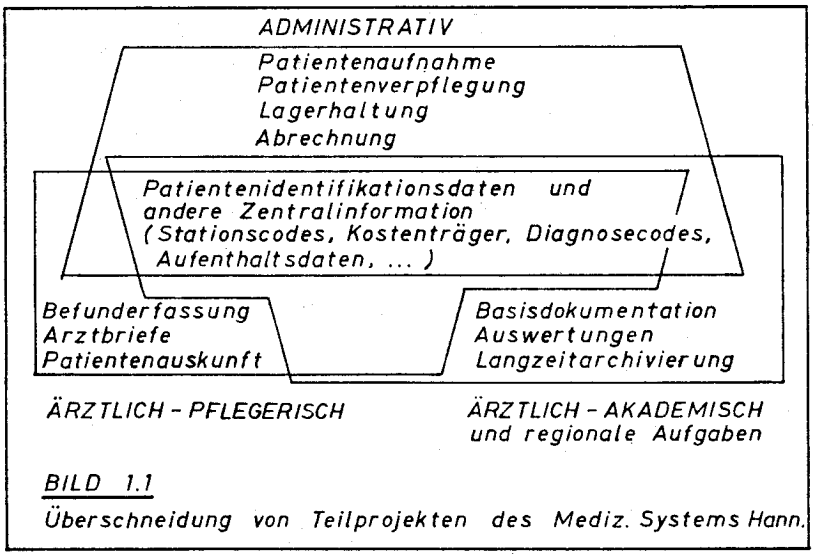


BILD 1.1

Überschneidung von Teilprojekten des Mediz. Systems Hann.

Die Komplexität des MSH wird durch die Darstellung einiger sich überschneidender Teilprojekte in Bild 1.1 deutlich. Neben einer Vielzahl von Einzelveröffentlichungen (wie z.B. 75) findet sich eine ausführliche Beschreibung in (65).

1.1.2.2 Probleme des Datenverarbeitungsgrundsystems

Unter DV-Grundsystem soll die Gesamtheit der für den normal vorausgeplanten Betriebsablauf des KIS erforderlichen Programme, Dateien und Datenbanken verstanden werden. Das DV-Grundsystem im MSH umfaßt z.B. Datenerfassungs-, Verarbeitungs-, Auskunfts-, Auswerte-, Dokumentations- und Archivierungssysteme für Patientendaten, aber auch Krankenhaus- bzw. Hochschulbetriebsdaten (Essenbestellung, Lagerhaltung, Haushaltsverbuchung, Angestellten- und Arbeitergehälter, Personaldaten, usw.).

Datenstrukturen, Programme und organisatorische Abläufe sind auf die Aufgaben des praktischen Einsatzes hin potimiert worden. Mit Hilfe der Programmiersprachen Assembler und PL/1 entstanden effiziente Prozeduren. Aus Effizienzgründen wird neben dem Datenbankverwaltungssystem IMS (Information Management System) (DMS, DBMS, DBVS) auch selbst programmierte Datenhaltung genutzt (11, 52, 62, 76, 103). Die damit entstehende physische Datenabhängigkeit der Datenverarbeitungsprogramme wird, wie auch z.B. die Redundanz von Information, bewußt in Kauf genommen.

Die Flexibilität des Systems (Daten- und Funktionsstruktur) wird durch hierarchische Strukturierung, genormte Datenschnittstellen und modularen Systemaufbau erreicht. So sind durch Neu- und Umprogrammierung grundsätzlich Systemänderungen möglich.
Es lassen sich die unerwünschten, wenn auch von den Betriebserfordernissen her nicht vermeidbaren Beschränkungen und Probleme im Grundsystem nennen, auf die weiter unten noch detaillierter eingegangen wird:

1. "ad hoc"-Anfragen und unvorhergesehene Problemstellungen
2. Programmierereffektivität und Nichtprogrammierereinsatz
3. Datenunabhängigkeit
4. Dauer der Durchführung von Systemmodifikationen
5. Datenrestruktuierungsaufwand.

1.1.2.3 Sprachkonzept zur Grundsystemergänzung

Umfassende Software, die die genannten Beschränkungen aufhebt, ist vom Aufwand her für das MSH nicht realisierbar. Programme für einzelne dieser Punkte müssen in sich sehr variabel sein, so daß die Eingabe von Steuerungsdaten über Parameter sehr kompliziert ist. Erfahrungen in dieser Hinsicht konnten mit dem sog. "Generellen Online Retrieval System" gemacht werden (72, 73). Die Eingaben waren praktisch nur vom Programmersteller selbst effektiv handhabbar. Mit dem sog. Trägersystem DADIMOPS im Bereich Befunderfassung (103) konnten zu Punkt 2 bis 4 der Problemliste im MSH weitere Erfahrungen gewonnen werden. Dieses System deckt aber nur einen kleinen Teil der Daten des MSH mit eingeschränkter Zielsetzung ab, so daß es noch kein die Subsysteme übergreifendes Datenverarbeitungsverfahren darstellt.

Es bietet sich deshalb als Ergänzung des Grundsystems die Definition und Implementierung einer Sprache zur Milderung der genannten Be-

schränkungen an. Dabei geht es nicht darum, eine Sprache zu erfinden, die sich von anderen durch neuartige Fähigkeiten unterscheidet, sondern darum, bestimmte Funktionen für Datenbankverarbeitungsprobleme effektiver programmieren zu können. Die Sprache soll in einfachster Weise Satz-Zugriffe in den hierarchischen Strukturen des MSH definieren können und die Verarbeitung von Datenelementen, aus denen im allgemeinen Datensätze zusammengesetzt sind, ohne Kenntnis der Lokalisation der Elemente in den Datensätzen und auch ohne Kenntnis der Formatumwandlungsregeln ermöglichen. Dabei sollen über ein Schnittstellenkonzept für Datensatzzugriffe Subsysteme erreichbar und flexibel verarbeitbar werden.
Mit diesen Zielstellungen, verbunden mit einfachen Ein-/Ausgabeoperationen, lassen sich die fünf Problempunkte mit vertretbarem Aufwand unterstützen.

Für die Ergänzung oder den Ersatz eines seit Jahren im Einsatz befindlichen Systems mit hierarchischen Datenbanken und eigener Datenhaltung, wie dem MSH, gibt es zur Zeit kein käufliches System, das die genannten Ziele einschließlich der Erweiterung auf eine relationale Sprache bietet.

Als für Nichtprogrammierer freundliche Sprache wird z.B. SEQUEL genannt (4, 5, 9, 13, 15, 45, 69, 86, 98). Diese Sprache wird, wenn sie als ausschließliche und damit "vollständige" Datenbankverarbeitungssprache konzipiert ist, in der Realisierung sehr kompliziert und aufwendig und benötigt ein relationales Datenverwaltungssystem, wie z.B. System R (5) demonstriert. Für ein hierarchisches Datenverwaltungssystem gibt es keine, dem System R vom Umfang entsprechende relationale Sprachdefinition, deren Übersetzung in effiziente Verarbeitungsalgorithmen angebbar wäre. Außerdem übersteigen die Entwicklungs- und Implementierungskosten für ein derartiges System den Rahmen eines Hochschulinstituts, dessen Hauptaufgabe Methodenentwicklung und der Betrieb eines KIS ist. Auch ist die zu erwartende Leistung oder Durchsatzrate (performance) solcher Systeme in konventionellen Rechnern speziellen Datenbankrechnern unterlegen (6).

Deshalb war es sinnvoll, eine Sprachhierarchie (Bild 1.2) anzustreben, die eine handhabbare und nach und nach realisierbare Abstraktion der Maschinenebene von unten nach oben durchführt. Der Verlust in höheren Schichten von prinzipiellen Fähigkeiten zu Gunsten einfacherer und leistungsfähigerer Sprachausdrücke und Sprachübersetzung wird in Kauf genommen.
Die sogenannte Schichtenarchitektur ist dabei ein allgemein anerkanntes und bewährtes Mittel der Systementwicklung (1, 51, 79, 82).

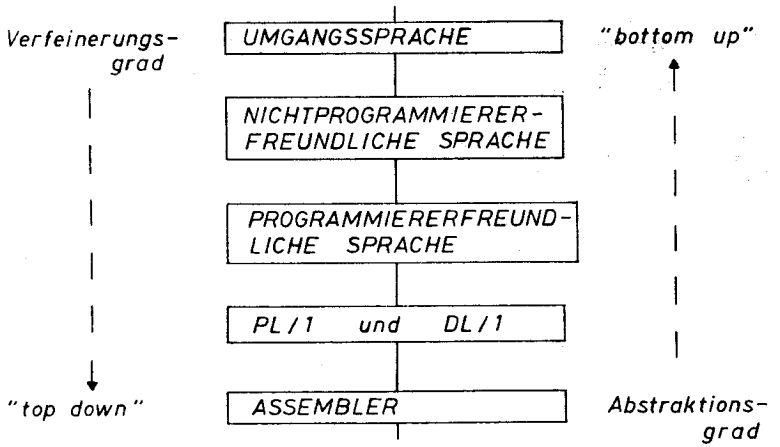


BILD 1.2

Sprachhierarchie als Abstraktion der Maschinenebene von unten nach oben in einem realisierten Schichtenmodell im Medizinischen System Hannover

Wenn deskriptiv und prozedural mit qualitativen Begriffen definiert wird, wie es z.B. Wedekind (98) beschreibt, dann ist eine Sprache deskriptiv, wenn in ihr nur das ausgedrückt werden muß, was das System tun soll. Dagegen ist eine Sprache prozedural, wenn dem System mitgeteilt werden muß, wie es was tun soll.
Die Schicht unterhalb der Umgangssprache soll in diesem Sinne eine deskriptive und SEQUEL ähnliche Sprache sein. Darunter soll eine mehr prozedurale Sprache liegen, die "fast alle" Datenbankverwaltungsfunktionen durchführen kann, jedoch von der physischen Implementierung der Daten unabhängig ist.
Diese Sprache wurde als eigenständige Sprache (self contained language) entworfen sowie implementiert und ist Hauptthema dieser Arbeit. Sie soll PDPL (Procedural Database Processing Language) genannt werden, weil sie eine der ablauforientierten Denkweise von Programmierern entgegenkommende Sprache ist. Sie hat einen beachtlichen Anteil nicht prozeduraler Sprachelemente, insbesondere für die Datenelementverarbeitung. Unter der PDPL kommt die Sprachschicht der üblichen Programmiersprachen mit ihren unterschiedlichen und speziellen Datenzugriffsarten und -konvertierungsmöglichkeiten.

Die weitgehende Automatisierung von Datenelementverarbeitung und Datensatzzugriffen, ohne das Datenbankverwaltungssystem (DBVS)

selbst anzusprechen, erfordert eine rechnerverarbeitbare Beschreibung der Datensätze, ihrer hierarchischen Abhängigkeiten und die Beschreibung der diese Sätze bildenden Datenelemente. Auch aus Gründen der Dokumentation der etwa 3000 Datenelemente und der Datenbankadministratoraufgaben wurde im MSH ein Daten- und Programmbeschreibungssystem (DAPRO) definiert, das auch die für die PDPL erforderlichen Informationen enthält (48, 72, 73). Dabei ist das Interne Schema des DAPRO, das für die PDPL-Übersetzung und die Ausführung der Sprachanweisungen benötigt wird, in wesentlichen Teilen im Rahmen dieser Arbeit entwickelt worden. Darauf soll hier aber nicht weiter eingegangen werden.

1.1.2.4 Leistungsfähigkeit der Sprachergänzung

Um das vorliegende System von seiner Leistung her zu beurteilen, muß eine Leistungsbewertung im Sinne der "performance evaluation" erfolgen (92, 94). Das Ziel einer Leistungsbewertung ist allgemein entweder

1) *die Leistungsfähigkeit zu kontrollieren und zu verbessern,*

2) *Systeme vergleichend zu klassifizieren (und auszuwählen)*

oder

3) *Systemänderungen in ihren Auswirkungen abzuschätzen.*

Ein allgemein anerkannter und meßbarer Begriff der Leistung (performance) fehlt (9, 106). Die Analogie Durchsatz zu Arbeit, Durchsatzrate zu tatsächlich erbrachter Leistung und Leistungsfähigkeit zu maximal möglicher (Nenn-)Leistung ist hilfreich; jedoch ist die Arbeitseinheit, in der z.B. Durchsatz von DV-Systemen gemessen werden soll, nicht allgemein gültig definierbar. Aus diesem Grund ist die Ermittlung der Leistungsfähigkeit eines Datenverarbeitungssystems oder -prozesses ein schwieriges Problem.

Systemleistung kann nur im Hinblick auf eine definierte Arbeitslast festgestellt werden. Das höchste Ziel der Systemanalyse ist die Angabe eines Modells der Leistungsfähigkeit (performance model), das den Zusammenhang von Leistungsmessungen mit der Systemstruktur und der Arbeitslast beschreibt (94).

Prinzipiell gibt es verschiedene Methoden der Leistungsermittlung (92):

- *Analytisches Modell*
- *Simulations- oder statistisches Modell*
- *Leistungsmessung*
- *Benchmarktest*
- *sog. Schreibtischanalyse*
- *Verfahrenskombination.*

Die Validierung von Modellergebnissen durch Vergleichsmessungen muß in jedem Fall erfolgen (32, 92, 94). Über den meßbaren Begriff der Leistung hinaus, der den Bereich der inneren Leistungsfähigkeit (in-

ternal efficiency) betrifft, gibt es eine äußere Leistungsfähigkeit (external efficiency), die eher qualitativen Charakter hat, z.B. die Benutzerfreundlichkeit eines Systems. Weitere Gesichtspunkte zum Problem Leistungsfähigkeit werden in Kapitel 1.2.3 und 3 besprochen. Für die Bewertung des vorgestellten Systems werden analytische Modelle für verschiedene Zugriffsmethoden des Datenbanksystems angegeben und durch Zugriffszeitmessungen validiert.

Der Leistungsvergleich zwischen einem relationalen Datenmanagementsystem auf einem Universalrechner (GPC, general purpose computer) und einem Datenbankrechner (DBC, data base computer) wurde auf der "Conference on very large data bases" 1978 in Berlin vorgelegt (6).

Der Vergleich eines hierarchischen Systems auf einem GPC mit einem relationalen DBC wurde im Rahmen dieser Arbeit über einen pragmatischen Weg versucht; dazu werden spezielle Arbeitslasten und Datenstrukturen des Medizinischen Systems Hannover ausgewählt und Antwortzeitvergleiche zwischen den beiden Systemen angestellt.

1.2 Bezug zur Datenbankforschung

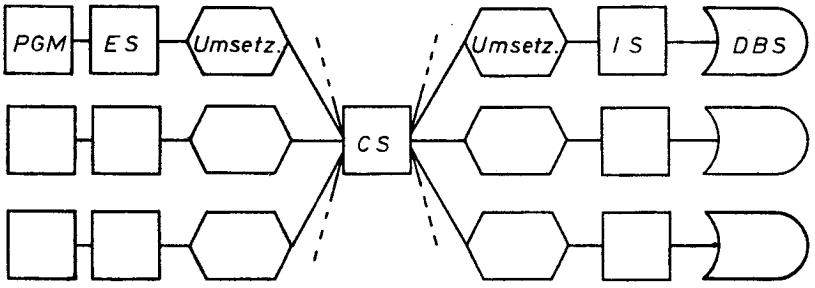
1.2.1 Datenmodelle

Zur Zeit des Entwurfes des MSH um 1971 wurde als produktionsreifes Datenmanagementsystem das hierarchische Informationsmanagement System IMS von IBM ausgewählt. In dieser Zeit erschien die bahnbrechende Publikation zum relationalen Datenmodell von Codd (15).

Der Streit, ob das relationale Datenmodell von Codd, das Netzwerkdatenmodell der CODASYL-Gruppe (14), das Graphenmodell DIAM von Senko oder das hierarchische Modell IMS von IBM (42) besser sind, wird heute nicht mehr geführt. Das Nebeneinander der Modelle wird akzeptiert, weil es möglich erscheint, eine fruchtbare Koexistenz zu gewährleisten, z.B. auf der Grundlage eines Schema-Modells, wie es vom Amerikanischen Institut für Nationale Standardisierung (ANSI) 1975 vorgeschlagen wurde (2). Blaser und Schmutz formulierten im selben Jahr in einer Bestandsaufnahme mit dem Titel "Datenbankforschung heute" , daß es "Probleme im DB-Bereich gibt, die mehr Energie und Aufmerksamkeit verdienen als Unterschiede zwischen Datenmodellen" (9).

Das Schema-Modell eröffnet nach Nijssen die Möglichkeit, über das Konzeptionelle Schema verschiedene Anwendersprachen, die im sog. Externen Schema definiert werden und verschiedene Datenmodellrealisierungen, die im Internen Schema beschrieben werden, zu verbinden. Er nennt das von ihm 1977 vorgestellte Modell deshalb auch Koexistenzmodell (58), siehe Bild 1.3. Auf der Benutzerseite ist das Externe Schema (ES) anzusiedeln, auf der Seite der Datenverwaltungssysteme das Interne Schema (IS). (Hier ist der Hinweis angebracht, daß die Schnittstelle DB-System als IS anzunehmen, wie im MSH geschehen, willkürlich ist, denkbar ist diese Schnittstelle auch auf einer tieferen Ebene, etwa der des Betriebssystems).

Klutentreter hat 1975 im Rahmen der Überlegungen zur Portabilität von Datenbanken in einem GMD-Bericht (49), Bild 1.4, eine Darstellung veröffentlicht, die vom Prinzip her auch bereits ein Schemaansatz ist.



PGM: Programm
DBS: Datenbanksystem
ES: Externes Schema
Umsetz.: Schemaumsetzung
CS: Konzeptionelles Schema
IS: Internes Schema

BILD 1.3

Koexistenzmodell von Nijssen als 'Drei-Schema-Modell'

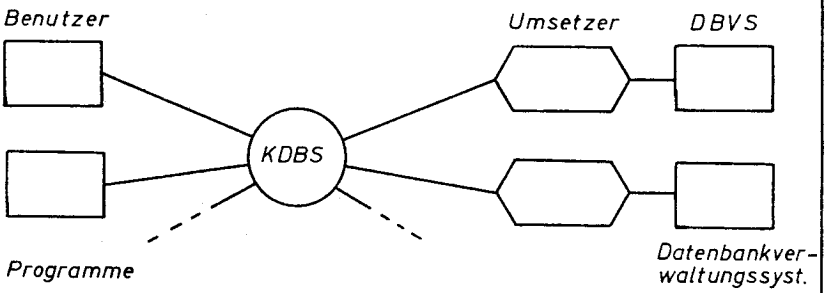


BILD 1.4

Kompatible Datenbankschnittstelle nach Klutentreter

Die sog. kompatible DB-Schnittstelle für komplexe Strukturen nach Bild 1.4 übernimmt Funktionen des Konzeptionellen Schemas (CS), nämlich die Verbindung ES---IS. Die sog. kompatible Anwendungsprogrammierung, ein die Bundesländer übergreifendes Projekt, hat Schnittstellen definiert für Datenbankmanagement, Datenkommunikation, einfache Dateien und Systemdateien und diese großteils realisieren lassen (10). In der Verwaltung wird in Niedersachsen die Schnittstelle für Datenbanken eingesetzt (7). Die Schnittstelle für Datenkommunikation, KDCS, findet sich zum Beispiel im Transaktionsmonitor UTM der Firma Siemens identisch wieder.

Die PDPL im MSH ist in einem ähnlichen Bild einordbar:

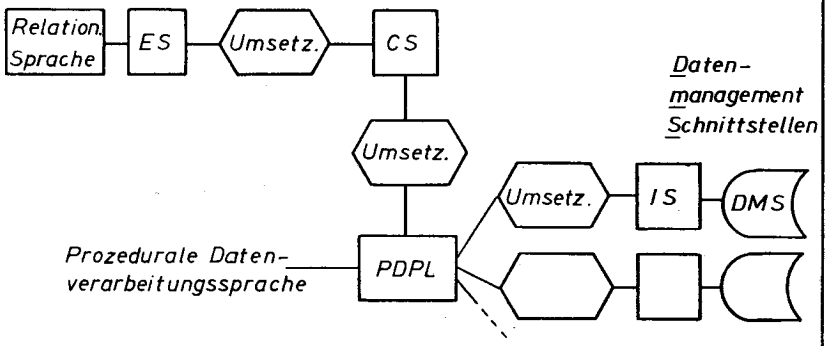


BILD 1.5

Einordnung der Funktion der PDPL in den Rahmen eines 'Drei Schema'-Modells für das Medizinische System Hannover. Dem Umsetzer auf die Internen Schemata der unterschiedlichen Datenverwaltungssysteme ist die die Datensatz- und Datenelementverarbeitung vereinheitlichende PDPL hinzugefügt.

Im MSH liegt dem Daten- und Programmbeschreibungssystem (72, 73) der 3-Schema-Ansatz zugrunde. Beschrieben werden Datensätze und Datenelemente mit ihren gegenseitigen Beziehungen. Die PDPL arbeitet auf dem Internen Schema und wird entweder direkt programmiert oder entsteht aufgrund der Übersetzung eines relationalen Externen Schemas.

Erforderliche Informationen für den Interpreter, wie z.B. die Datenverwaltungssystemeigenschaften für den Datensatzzugriff, Format-

konvertierungsregeln oder Plausibilitätsbedingungen für Datenelementinhalte entstammen dem Datenbeschreibungssystem. Darauf beruht die physische und zum Teil auch logische Datenunabhängigkeit.

1.2.2 Vergleichbare Zielsetzungen

Ähnliche Zielsetzungen wie im MSH wurden 1975 von Frasson (22) mit seiner Sprache Alpha für IMS-Strukturen vorgestellt, die jedoch durch Beschränkung auf die Datendefinitionssprache von IMS eine sehr eingeschränkte Einsatzbreite hat.

1977 beschrieb Risch (3, 70) einen Quellcodegenerator, der IMS-DB-Verarbeitung dadurch unterstützt, daß aufgrund einer SEQUEL-ähnlichen Eingabesprache COBOL-Code für die Datenbankzugriffe generiert wurde.

Dubien et.al. (17) stellten 1977 ein 3-Schema-Modell für das DB-System TOTAL vor, zu dem zu der Zeit noch keine Sprachdefinition erstellt war.

Wildgrube hat auf der GI Jahrestagung 1976 angekündigt, daß für das DBTG-Datenmanagementsystem UDS in der Version V2 eine relationenorientierte Sprache mit natürlich-sprachlichen Elementen auf der Basis eines Koexistenzmodells geplant ist.

Böhm vom Universitätskrankenhaus Eppendorf (Hamburg) berichtete Oktober 1978 im Informatik Seminar an der MHH über ein relationales Datenverwaltungssystem mit SEQUEL-ähnlicher Sprache auf der Basis des Systems ADABAS. Hier war wegen der Nähe des Datenverwaltungssystems zu relationalen Systemen der Schemagedanke nicht sehr ausgeprägt.

Datenbankverwaltungssysteme bieten heute in der Regel Abfragesprachen (Querysprachen) herstellerseitig an. Diese sollen hier nicht alle aufgeführt werden. Die Sprache IQF (43) z.B. ist eine solche "self contained language", die es erlaubt, im Dialog Anfragen über die Daten einer Datenbank des IMS-Systems mit einem Abfrageformalismus anzusprechen. Im Gegensatz zur PDPL des MSH hat sie aber z.B. keine Zwillingssegmentverarbeitung, keinen Änderungsdienst und ermöglicht auch nicht das Verbinden unterschiedlicher hierarchischer Pfade beliebiger Datenverwaltungssysteme. So hat die PDPL, auch durch den relationalen Überbau, eine dagegen wesentlich erweiterte Zielsetzung.

1.2.3 Zur Leistungsbewertung

Für die Durchführung der Bewertung der Leistungsfähigkeit des Konzepts "Erstellung einer Datenbankverarbeitungssprache zur Ergänzung der Verarbeitungsprogramme eines KIS", muß das Problem der Leistungsbewertung noch detaillierter dargestellt werden.

Zum einen ist Leistungsfähigkeit (performance) ein Maß dafür, wie effektiv ein System eine spezifische Anwendung realisiert

(effectiveness), und zum anderen ist sie ein Maß für die Effizienz (efficiency) bei der Durchführung dieser Anwendung. Während die äußere Effektivität (external) die Frage beantwortet, wie gut ein System den Benutzer in die Lage versetzt, das zu tun, was er mit dem System vorhat zu tun, beantwortet die innere Effizienz (internal) die Frage, wie gut oder schnell das System die gegebenen Aufgaben erledigt.
So fallen unter die äußere Effektivität (47)

1. *die Klarheit der Definition, welche Probleme vom System gelöst werden sollen und*
2. *die Leichtigkeit, wie diese Problemlösungen mit Hilfe des Systems definierbar sind.*

Die innere Effizienz (47) dagegen umfaßt die Effizienz, mit welcher die rechnerinterne Datenstruktur zur Problemlösung verarbeitet wird. Man kann auch versuchen, die Leistungsfähigkeit geschichtlich zu begreifen (32):

- *"Performance früher" als Geräteproduktivität (device productivity)*
- *"Performance heute" als Systemproduktivität (system productivity)*
- *"Performance morgen" als Anwenderproduktivität (people productivity).*

In Schlagwörtern bedeuten diese drei Begriffe auch:

- *Geschwindigkeit, Leistungsvermögen, Anwenderfreundlichkeit (106)*

oder

- *Ausführungseffizienz, Implementierungseffizienz, Benutzereffektivität (79).*

Hill (32) bezieht das eigentliche Leistungsproblem auf die Flaschenhälse in einem System, denn nach Beseitigung des einen gerät man an den nächsten usw.: *"Because ... there always is a next one."* Seine Definition, ungeachtet der drei historischen Aspekte, lautet: Leistungsfähigkeit ist die Qualität der Beherrschung des Einsatzes von Betriebsmitteln unter einem definierten Arbeitsprofil. Ziel ist die Ausgewogenheit zwischen vergeudeten (unterbelegten) und knappen (überbelegten) Betriebsmitteln.

Die Quantifizierung der Ausführungseffizienz wird sehr ausführlich bei Stimler (92) und Svobodova (94) diskutiert. Es wird deutlich, daß das zu untersuchende System und das Ziel der Aussagen zur Leistungsfähigkeit festgelegt werden müssen, da sich hieraus der Aufwand auf Seite der Leistungskenngrößen und der Detaillierung der Modellbildung ergibt. Wichtig erscheint die Feststellung, daß der rekursive Begriff "System" zu einer genauen Definition desselben zwingt (94):

> *"System ist eine Struktur, die aus Elementen besteht, die aufgrund festgelegter Regeln miteinander in Beziehung tre-*

ten. Ein Element eines Systems kann seinerseits ein System sein, welches durch Elemente auf einer tiefer liegenden Schicht gebildet wird. Es wird nötig, zu entscheiden, in welcher Schicht ein System analysiert werden soll. Die Elemente werden dann als nicht weiter gliederbare Teile des Systems angesehen."

Zur Problematik der Modellbildung werden weitere Anmerkungen bei der Betrachtung der Methode der Systemsimulation im Kapitel 3.2.1 erforderlich.

Die Publikationen zur Leistungsbewertung befassen sich mit Ausnahme der Untersuchungen zur Geräteproduktivität (devices) damit, den Rechner insgesamt leistungsmäßig zu beurteilen. Bei einzelnen Programmsystemen oder Rechnerteilen liegt es nahe, gleiche Methoden zur Beurteilung der Leistungsfähigkeit anzuwenden, weil die Problematik der Nutzung von Betriebsmitteln und der Definition des Systems, seiner Arbeitslast und seines Durchsatzes vergleichbar sind.

1.3 Inhalt der Folgekapitel

Im Kapitel 2 wird das Konzept der Syntax und die Implementierung des Gesamtsystems beschrieben. Mit nur neun Anweisungstypen und einer eingeschränkten, aber anschaulichen und im Vorgehen programmiererfreundlichen Methode der Verknüpfung dieser Sprachelemente, lassen sich komplexe Datenverarbeitungsaufträge definieren. Bei der Implementierung wurden Erfahrungen mit der Methode der strukturierten Programmierung und HIPO (40) (Hierarchy plus Input, Process, Output) gewonnen.

Im Kapitel 3 wird versucht, eine Systembewertung durchzuführen. Als Hauptkriterium der Leistungsfähigkeit wird das Antwortzeitverhalten bei vorgegebener Datenstruktur und Arbeitslast definiert. Weitere Kriterien, wie Programmierereffektivität und Speicherplatzkosten werden angeführt. Die PDPL ist ein Konzept, das keine Entlastung von CPU-Betriebsmitteln ermöglicht. Neuere Methoden zur Datenbankverarbeitung versuchen durch Verlagerung von Intelligenz in die Peripherie den heute festzustellenden "Flaschenhals" CPU zu entlasten. Zu diesen Konzepten zählt das Projekt "Suchrechner" an der Technischen Universität Braunschweig (46, 53, 54, 105). Im Vergleich der Leistungsfähigkeit mit diesem Datenbankrechner sollen Stärken und Schwächen beider Methoden zur Verarbeitung von Datenbanken relativ zueinander ermittelt werden.

In Kapitel 4 werden die Ergebnisse der vorliegenden Arbeit nach den wichtigsten Gesichtspunkten diskutiert und zusammengefaßt. Eine Tabelle mit vier qualitativen Kriterien, die die Art der Arbeitslast charakterisieren, zeigt Gesetzmäßigkeiten für die Größenordnung des Verhältnisses der Zugriffszeiten beider Verfahren.

Im Kapitel 5 befindet sich das Literaturverzeichnis.

In Kapitel 6 ist die Definition der Sprachsyntax für die Verarbeitungssprache PDPL sowie für den Suchmodul-Assembler des Datenbankrechners beigefügt.

2. SYNTAXKONZEPT UND SPRACHIMPLEMENTIERUNG

2.1 Pfad-orientiertes Zugriffs- und Verarbeitungsprinzip

Die Sprache wird in Kapitel 6 formal dokumentiert. Das Konzept der Sprache sieht vor, eine Einheit aus Datensatzzugriff und zugeordneter Datenelementverarbeitung als Zugriffspfad zu definieren, was von der sonst üblichen Definition eines Zugriffspfades abweicht (98). Ein vollständiger Datenverarbeitungsauftrag, analog dem Programm in einer Programmiersprache, wird aus mehreren Zugriffspfaden gebildet, die über die verfügbare Kontrollogik verbunden sind. Zur Veranschaulichung ist eine mögliche Zugriffspfadsequenz von drei Pfaden in Bild 2.1 dargestellt.

Der Interpreter durchläuft, falls er nicht durch Kontrollogik anders gezwungen wird, in einer vorgegebenen Ausführungsfolge den Zugriffspfad. Die in einer festgelegten Reihenfolge zur Verfügung stehenden Verarbeitungsfunktionen werden durch die Sprachanweisungen aktiviert oder nicht.

Die Sprachanweisungen beziehen sich auf die drei Aufgaben:

1. *Datensatzzugriff*
2. *Datenelementverarbeitung*
3. *Kontrollogik (Ablaufsteuerung)*

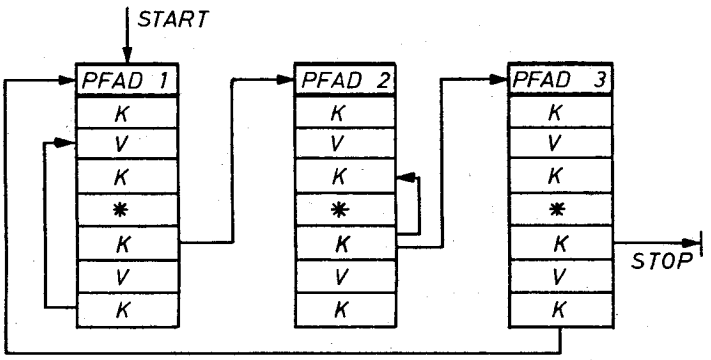


BILD 2.1

Darstellung des Pfad-orientierten Verarbeitungskonzeptes

2.2 Sprachanweisungen

Sprachanweisungen bestehen aus einer dreibuchstabigen Kennung, dem Anweisungstyp, und einem Klammerausdruck, dem Aussehen nach formal einer Parameterliste von Prozeduren entsprechend. Sprachanweisungen werden in sinnvoller Reihenfolge aneinandergefügt zum Datenverarbeitungsauftrag.

2.2.1 Datensatzzugriff

Der Datensatzzugriff erfordert die Angabe des Datensatznamens als Referenz zum Internen Schema (SFS = specify segment, Seite 147) und eine nähere Bezeichnung der Art des Datensatzzugriffs (Art des Lesevorganges, ACC = access definition, Seite 148).

```
-----------------------------------------------------------------
I  SFS (lfd. Zugriffspfadnr., Datensatzname                     I
I  ACC (Zugriffsfunktion, Schlüsseloperatordefinition)          I
-----------------------------------------------------------------
```

Zugriffsfunktionen sind:

Get unique, get next, get next within parents, insert, usw.

Schlüsseloperat**oren** sind:

> , = , < , <= , >= , ≠

2.2.2 Datenelementverarbeitung

Die Sprachdefinition wurde mit einem Datenelementdeklarationsteil ausgestattet, was im Prinzip nicht nötig gewesen wäre, weil ein Datenelement durch seinen eindeutigen Namen aus dem Internen Schema hinreichend definiert ist. Aus Gründen der Einsparung von Schreibarbeit und der Möglichkeit der Mehrfachreservierung von Platz für dasselbe Datenelement für den Programmierer war jedoch der Deklarationsteil sinnvoll (DDC = data declaration, Seite 146).

```
-----------------------------------------------------------------
I DDC (Datenelementnamenliste)                                  I
-----------------------------------------------------------------
```

Der Datenelementverarbeitungsteil bezieht sich auf:

1. *Bewegen (Transport) von Datenelementwerten in die durch den Deklarationsteil reservierten Plätze hinein (IDB = Into Data Buffer) oder aus ihnen heraus (ODB = Out of Data Buffer) (Seite 149 und Seite 150),*
2. *Manipulation von Datenelementwerten (CPT=Compute , Seite 152) und*
3. *Definition der an Datenelementinhalte zu stellenden Bedingungen (WHR = Where, Seite 151).*

IDB (Transport von woher, Datenelementreferenzliste)
ODB (Transport wohin, Datenelementreferenzliste)
CPT (Verarbeitungsanweisung)
WHR (Disjunktion von Gruppen mit jeweils konjunktiv verbundenen Prädikaten)

2.2.3 Kontrollogik

Der prozedurale Anteil der Sprache wird im wesentlichen durch die Kontrollogikanteile bestimmt, die unter anderem auch bedingte Ausführungsverzweigungen ermöglichen. Eine ausführliche Dokumentation der einzelnen Kommandos befindet sich in Kapitel 6 (Seiten 137 bis 143).
Die Definition geschieht durch die sog. Kommandoanweisung:

CMD (Kommando)
SCB (Kommando)

Kommandos können Teil der Datenverarbeitungsaufgabe selbst sein, oder der unmittelbaren Steuerung des Dialoges außerhalb der Datenverarbeitungsaufgabe dienen. Im unmittelbaren Dialog im System GURS entfällt die Kennzeichnung eines Kommandos durch CMD, da die Eingabeprüfroutine CHECK auf Kommandos selbst prüft.

Das Dialogprogramm, das die Definition und Durchführung von Verarbeitungsaufträgen in der Sprache PDPL ermöglicht, akzeptiert Grundeinstellungen des Programmablaufs in Form von Kommandos, die ohne die Kennzeichnung CMD (...) eingegeben werden.
So wird z.B: Verarbeitungsunterbrechung nach jeder Bildschirm- bzw. Druckerausgabe durch direkte Eingabe von BRAKE voreingestellt. Dadurch erhält man die Möglichkeit des Dialogeingriffs während der Durchführung des durch die PDPL definierten Verarbeitungsauftrages jeweils nach der Ausgabe von Ergebnissen. Keine Unterbrechung wird mit NOBRAKE erreicht, was im Stapelbetrieb sinnvoll ist. Die vorgegebene Einstellung (default) ist in diesem Fall BRAKE, so daß Grundeinstellungen nicht unbedingt durchzuführen sind.

Zugriffspfadverzweigungen (-umschaltungen) erfolgen z.B. durch CMD (BCH, Pfadnummer).

Kommandos werden unbedingt oder in drei Fällen bedingt gegeben:

CMD (WHR, 0, Kommando) = Kommandoausführung bei Ergebnis "false" der mit WHR (...) gestellten Bedingung im Datenelementverarbeitungsteil. Eine ausführliche Dokumentation erfolgt auf Seite 153.

CMD (CPT, Kommando) = Kommandoausführung im Falle eines Verarbeitungsfehlers in einer davorliegenden CPT-Anweisung, z.B. bei Konversionsfehler. Die Fehlerbedingungen werden nicht näher differen-

ziert und auch nicht auf eine bestimmte CPT-Anweisung, sondern auf die gesamte im PDPL-Auftrag unmittelbar davor liegende Gruppe von CPT-Anweisungen bezogen.

SCB (XX, Kommando) = Kommandoausführung nach Aufruf eines Datenverwaltungssystems je nach dem, welcher Statuscode = XX von diesem nach dem Datensatzzugriff zurückgegeben wurde (Einzelheiten siehe Seite 154).

2.2.4 Darstellung der Sprachelemente

Bild 2.2 zeigt alle möglichen Sprachelemente in ihrer vorgegebenen Ausführungsreihenfolge je Zugriffspfad. In der Darstellung wird auch das Zugriffspfadkonzept verdeutlicht. Die Reihenfolge der Anweisungstypen ist für jeden Zugriffspfad fest vorgegeben. Ob sie jedoch aktiviert werden oder nicht, hängt von der Definition im Suchauftrag ab. Damit die Anzahl der Anweisungstypen nicht zu groß wird, wurden noch zwei Positionstypen (SSA und CAL) eingeführt. Die Elemente SSA (Pfadbeginn) und CAL (Datensatzzugriff) haben Platzhalterfunktionen, damit die Ausführung z.B. von Kommandos mit dem gleichen Kennungstyp CMD zu verschiedenen Zeitpunkten des Durchlaufes definiert werden kann:

- *einmalig zu Beginn des Pfaddurchlaufes (vor SSA)*
- *vor Durchführung des Datensatzzugriffes (nach SSA, vor CAL)*
- *nach Durchführung des Datensatzzugriffes (nach CAL).*

Wie in Bild 2.13 noch erläutert wird, stellt ein einzelner Pfad standardmäßig eine Verarbeitungsschleife dar, weil nach dem letzten CMD wieder zurückverzweigt wird zu SSA.

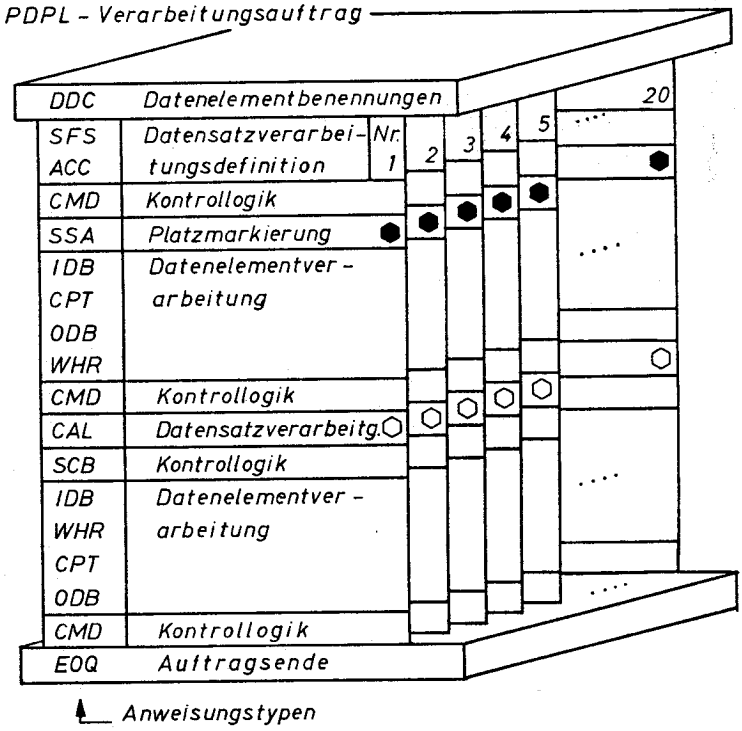


BILD 2.2

Der Datenverarbeitungsauftrag wird als Folge von Zugriffspfaden definiert. Jeder der bis 20 Zugriffspfade besteht aus einer vorgegebenen Sequenz von Verarbeitungsfunktionen, die mit den Anweisungstypen der PDPL korrespondieren und je nach Auftrag aktiviert werden oder nicht.

2.3 Sprachimplementierung

2.3.1 Gliederung in Funktionen

Das Hauptprogramm GURS (Generelles Update- und Retrievalsystem) enthält die Funktionen, um Datenverarbeitungsaufträge im PDPL-Format zu definieren und auszuführen. Das Hauptprogramm ist in sieben Funktionen gegliedert, die zum Teil noch weiter horizontal und vertikal in Moduln aufgegliedert werden können. Die sieben Funktionen sind:

1. *Makro für Dialog und Stapelversion*
2. *Dialogsteuerung*
3. *Variableninitialisierung*
4. *Erklärungsbilder*
5. *Bearbeitung von Datendefinitionen*
6. *Syntaxanalyse und Sprachübersetzung*
7. *Interpreter*

Eine tiefe hierarchische Gliederung der Funktionen in Moduln (hierarchical decomposition) war nicht erforderlich. Die Hauptgliederung bis auf die Funktion Nr. 6 bestand aus einer ebenen Aufteilung der Funktionen in Hauptprogrammoduln , den hier sog. Programmblöcken des Hauptprogramms (modular decomposition). Die Funktionsgliederung enthält Bild 2.3. Der Interpreter soll auch als Zugriffspfad- oder Pfadmaschine bezeichnet werden.

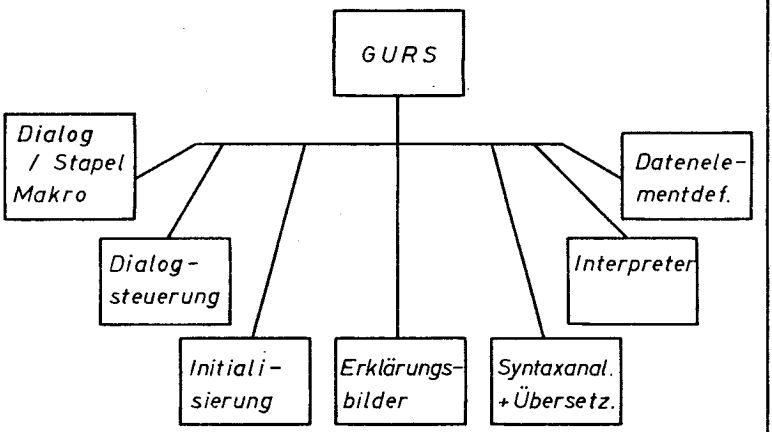
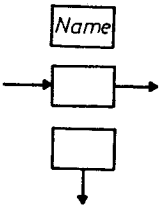


BILD 2.3

Die Grobfunktionsgliederung des Generellen Update- und Retrievalsystems GURS

2.3.1.1 Hauptprogrammoduln (Horizontale Gliederung)

Die horizontale Gliederung der Programmfunktionen in Programmblöcke ist im Bild 2.4 dargestellt. Es wird versucht, die Verzweigungsmöglichkeiten zu zeigen, ohne auf die Programmblockinhalte im Detail einzugehen. Die Eingabetestroutine CHECK, die eigentlich zur vertikalen Gliederung zu rechnen ist, wird deshalb als Hauptprogrammblock angegeben, weil sie Hilfsfunktionen im Dialog selbständig abwickelt, wie z.B. das Wegschreiben eines Datenverarbeitungsauftrages auf den Speicher. Eine Zeichenerklärung für Bild 2.4 soll hier gegeben werden:



Programmblock (Modul), ein Eingang, ein normaler Ausgang, eventuell zusätzliche Fehler- oder Nebenausgänge.

Normaler Ein- und Ausgang

Neben- oder Fehlerausgang

Unbedingter Programmverlauf

Pfeil mit Knoten: Ablauf mit Dialogeingabeunterbrechung und Aufruf der Eingabeprüfroutine CHECK der Dialogsteuerung, Rückkehr in den Knoten, falls CHECK nicht anders entscheidet.

Kommt zur Dialogsteuerung aus Knoten im Symbol:

Geht aus der Dialogsteuerung zurück in den Knoten des Symbols:

N

Geht aufgrund der Eingabetestroutine und des Inhalts der Eingabevariablen zu einem Programmblock, der durch die Eingabe selbst vorbestimmt ist (Kommando z.B.)

CMD

Aufruf der Eingabetestroutine CHECK nicht von der Dialogsteuerung aus, sondern von Programmblöcken aus, kehrt je nach Inhalt der Dialogvariablen zum Aufruf zurück oder zu Symbol:

BCH,#

Bedeutet Pfadumschaltung auf Pfadnummer = # des Verarbeitungsauftrages und Verzweigen zu Programmblock QUE.

CMD

Verzweigung hierher nur über Kommandoeingabe möglich (CHECK).

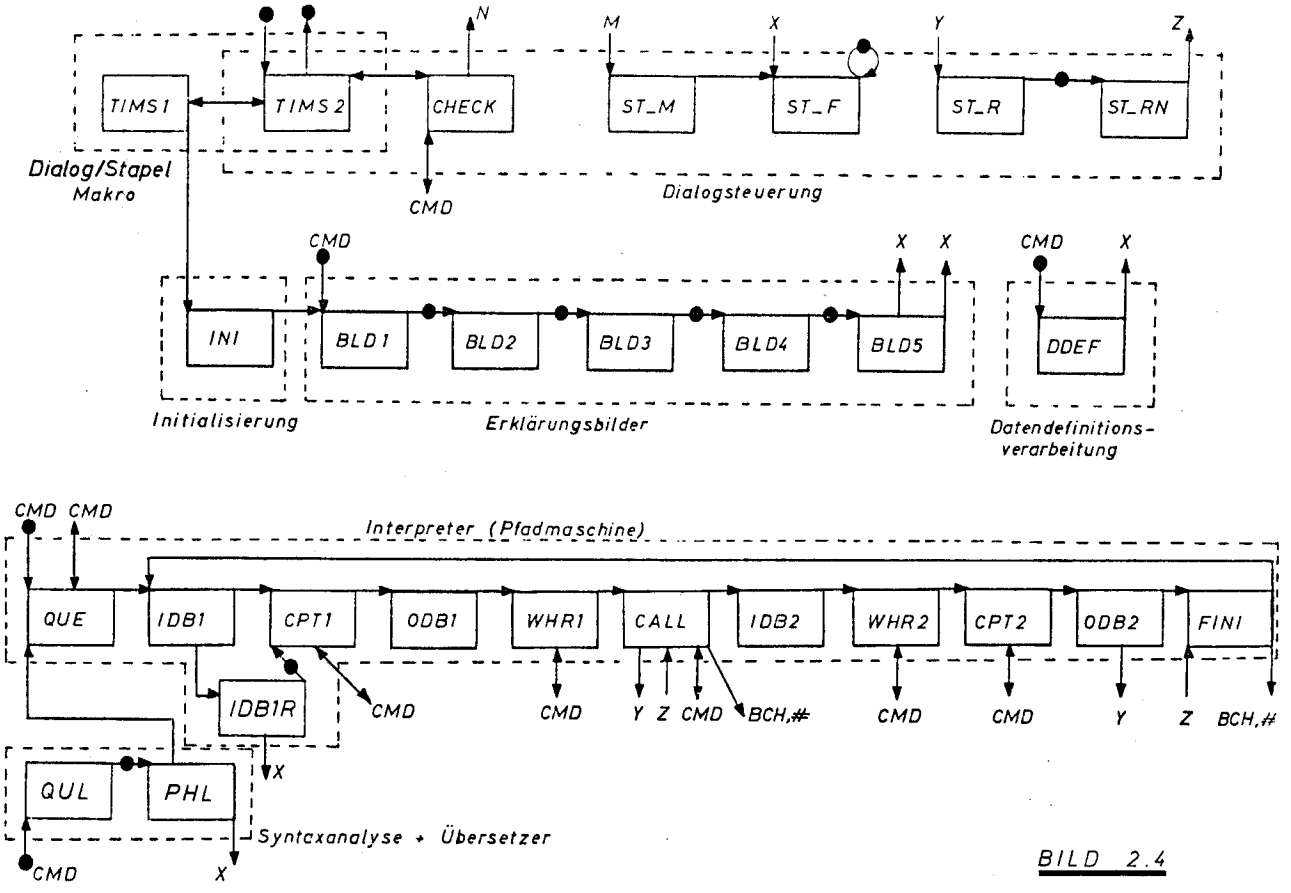

N
M
X
Y
Z
TIMS1
TIMS2
CHECK
ST_M
ST_F
ST_R
ST_RN
Dialog/Stapel
Makro
CMD
Dialogsteuerung
CMD
X
X
CMD
X
INI
BLD1
BLD2
BLD3
BLD4
BLD5
DDEF
Initialisierung
Erklärungsbilder
Datendefinitions-
verarbeitung
CMD CMD
Interpreter (Pfadmaschine)
QUE
IDB1
CPT1
ODB1
WHR1
CALL
IDB2
WHR2
CPT2
ODB2
FINI
IDB1R
CMD
CMD
Y
Z
CMD
BCH,#
CMD
CMD
Y
Z
BCH,#
X
QUL
PHL
Syntaxanalyse + Übersetzer
CMD
X

BILD 2.4

Die Aufgaben der Programmblöcke nach Bild 2.4, von denen ausgewählte weiter unten noch eingehender beschrieben werden, sollen kurz vorgestellt werden:

TIMS1: *Das Teleprocessing Interface Makro System des MSH sorgt für die Verwaltung mehrerer Bildschirmbenutzer in einem Anwendungsprogramm und für die Bildschirm- und Dateiverbindung zum Anwendungsprogramm (68). In der Stapelversion von GURS entfällt dieser Teil.*

TIMS2: *Der Teil des Teleprocessing Interface Makrosystems, der sich auf die Behandlung von Dialogunterbrechungen im Anwendungsprogramm bezieht: Einholen der Bildschirmnachricht, Aufruf der Eingabetestroutine und Rückverzweigen in das Anwendungsprogramm. In der Stapelversion von GURS besteht* TIMS2 *aus einem Programmabschnitt, der einen "Dialog" durch Abrufen/Ausgeben von Nachrichten aus/in der/die Systemeingabe/-ausgabedatei Sysin/Sysout abwickelt und ebenfalls die Prozedur CHECK aufruft.*

ST_M : *Ausgabe einer Meldung und Verzweigen zu* ST_F

ST_F : *Sog. Fehler- und Nachrichtenauflaufpunkt, Verlassen dieses Programmteiles nur über Kommandos.*

ST_R,ST_RN:
Spezialzweck: Erzeugen einer Dialogeingabeunterbrechung und Rückkehr zu einem vor dem Verzweigen zu ST_R *im Programm festzulegenden Rückkehrblocknamen.*

INI : *Programmvariableninitialisierung*

BLD1,...,BLD5:
Erklärungssequenz von Bildern, diese Sequenz wird im Dialog über das Kommando "?" erreicht.

QUL : *Dient der Eingabe eines Datenverarbeitungsauftrages in PDPL-Notation oder als relationale Datenanfrage (Query).*

PHL : *Stellt den Übersetzer relational in PDPL, die Syntaxanalyse der PDPL und die Übersetzung in eine für den Interpreter günstig zu verarbeitende Datenstruktur zur Verfügung. Im Falle eines vorangegangenen QCAL-Kommandos (Kapitel 6) verzweigt die Verarbeitung zur Auftragsverarbeitung, anderenfalls zu* ST_F.

QUE : *Initialisierung von Variablen und Durchführung von Kommandos. Verzweigen zu IDB1, dem Beginn einer Sequenz von Programmblöcken (Interpreter, 2.3.4) bis FINI, von dem aus zurückverzweigt wird an den Anfang dieser Sequenz, IDB1. Die Bedeutung dieser Einzelblöcke des Interpreters ist in Bild 2.13 angegeben. Anfang dieser Sequenz, IDB1. Einige der Blöcke der Sequenz ermöglichen der Ablaufkontrollogik eingeschränkte Verzweigungen, auch aus dem aktiven Pfad zu einem anderen.*

2.3.1.2 Unterprogramme (Vertikale Gliederung)

Untergliedert sind nur die Funktionen Syntax und Übersetzer (PHL), Bild 2.5, und einzelne Programmblöcke des Interpreters, Bild 2.6.

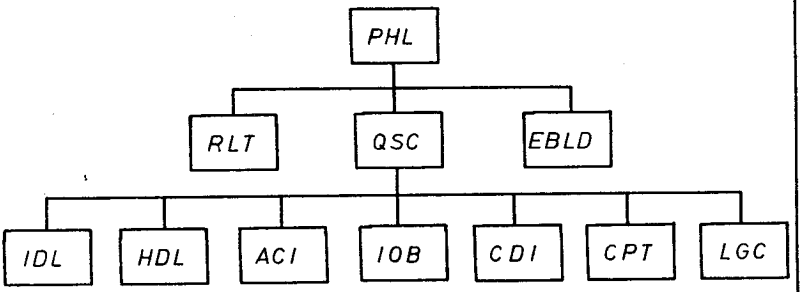


BILD 2.5

Untergliederung des Hauptprogrammblockes PHL 'Syntax und Übersetzer' in Unterprogramme (Vertikale Modularisierung)

Die Bedeutung der Moduln des Übersetzers nach Bild 2.5 sind:

RLT Übersetzung der relationalen Abfragesprache in die PDPL

QSC Syntaxprüfung des Verarbeitungsauftrages (Feststellen der Sprachanweisungen, ihrer Bedeutung, ihrer Zugriffspfadzugehörigkeit und ihrer relativen Stellung zueinander).

Übersetzung der einzelnen Sprachanweisungen in eine für den Interpreter verarbeitbare Datenstruktur:

IDL : *DDC-Anweisungen*
HDL : *SFS-Anweisungen*
ACI : *ACC-Anweisungen*
IOB : *IDB/ODB-Anweisungen*
CDI : *CMD/SCB-Anweisungen*
CPT : *CPT-Anweisungen*
LGC : *WHR-Anweisungen*
EBLD: *Generierung und Zwischenspeicherung von Bildschirmmasken für Eingaben aufgrund der Anweisung IDB (EX, ...).*

Die Bedeutung der Unterprogramme der Pfadmaschine nach Bild 2.6 sind:

PUT: *Erzeugen einer Ausgabe auf dem Bildschirm bzw. Drucker*

CPTPRC: *Durchführung der CPT-Anweisung und Bereitstellung der Ergebnisse im Datenelementspeicher und Setzen einer Fehlervariablen*

EXTINT: *Umwandlung der internen Darstellung eines Datenelementes in seine externe Repräsentation.*

INTEXT: *Umgekehrt zu* EXTINT

DMS$_1$,...,DMSn: *Speziell programmierte Schnittstellen für Datenmanagementsysteme. Zur Zeit sind verfügbar: sequentielle Dateien, Direktzugriffsdateien, PATSY (Patientensummary (76), spezielles DMS), alle IMS-Datenbanken, ein Tabellenmanager (Vorläufer für Verarbeitung von Zwischenergebnissen), Patientennamensinvertierung (spezielle Datenhaltung) und Diagnoseninvertierung (spezielle Datenhaltung).*

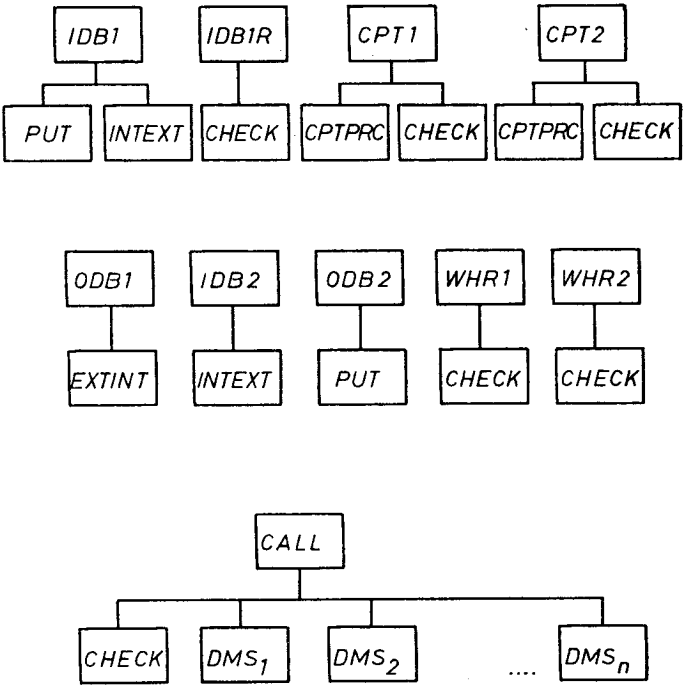


BILD 2.6

Untergliederung der Hauptprogrammblöcke des 'Interpreters' in Unterprogramme

2.3.2 Programmrealisierung

In diesem Kapitel wird die Beschreibung der Prinzipien und weniger die der Einzelheiten in Form einer Programmdokumentation vorgenommen.

2.3.2.1 Allgemeines

Zu Bild 2.2 bzw. 2.4 ist zu bemerken, daß nur der Datenzugriffsteil, repräsentiert durch den Sprachtyp und Platzhalter CAL() bzw. den Programmblock CALL , eine speziell programmierte Schnittstelle darstellt. Alle anderen Programmblöcke sind für alle Datenmanagementschnittstellen einheitlich realisiert. Dadurch ist der Übersetzer "relational in PDPL" nicht mit DMS-Spezialitäten zu belasten. Die relationale Schnittstelle soll nicht betrachtet werden, sie wurde in (48) beschrieben.

Im Sinne des "software engeneering" (1, 12, 16, 18, 19, 24, 26, 29, 33, 78, 80, 82) wurde das Programm

- durch eine mit Hilfe des Vorübersetzers (precompiler) eingesetzte Programmablaufsteuerung der horizontal gegliederten Funktionen,
- die vertikale Gliederung in Form einer flachen hierarchischen Dekomposition,
- die Beschränkung des Umfangs der Programmblöcke und Unterprogramme auf eine überschaubar kleine Menge Code und
- durch den Einsatz von Dokumentations- und Programmiermethodik

in gut wartbarer Form erstellt. Da die Übersetzung wegen des eingeschränkten Pfadkonzeptes einfach zu erstellen war, konnte auf rekursive Technik verzichtet werden.

Als Besonderheit der Dokumentation und Entwicklung wurde statt der herkömmlichen Programmablaufpläne das Verfahren HIPO (Hierarchy plus Input Process Output) (40) gewählt. Dabei wurde der bei HIPO den Prozess beschreibende Teil durch Struktogramme nach Nassi/Shneidermann (57) ersetzt. Diese mußten sich ihrerseits grafische Änderungen gefallen lassen, damit der Aufwand der Erstellung erleichtert wurde (keine Schräglinien).

Die verwendete Programmiermethode ist im Groben als strukturiert zu bezeichnen. Die Restriktion nach Nassi/Shneidermann auf Strukturbausteine mit nur einem Ein- und Ausgang erscheint als zu einschränkend, da dadurch zu kleine Dokumentationseinheiten entstehen. Das Verbot von GOTO-Anweisungen sollte innerhalb der überschaubaren Dokumentationseinheiten nicht gelten, da sonst unnötig tiefe Codeschachtelungen und das Einführen unnötiger Schaltvariablen für die Schleifenendebedingungen die Folge sind.

Als Beispiel für die HIPO Dokumentation in Verbindung mit Nassi-Shneidermann-Diagrammen sind in den Bildern 2.7 bis 2.9 die aufeinanderfolgenden Hauptprogrammblöcke CPT1, ODB1 und WHR1 ausgewählt worden. Schwarz ausgefüllte Pfeile bedeuten Programmablauf, nicht ausgefüllte Pfeile bedeuten Datenfluß. Jedes Diagramm hat nur einen Haupteingang und -ausgang. Daneben sind ganz wenige Neben- oder Fehlerausgänge vor-

gesehen. Im Fehlerfall wird der Fehlerauflaufpunkt erreicht. Die durch Aufruf der Eingabetestroutine CHECK erreichten Programmpunkte sind nicht vorhersehbar (durch das Symbol X angedeutet).

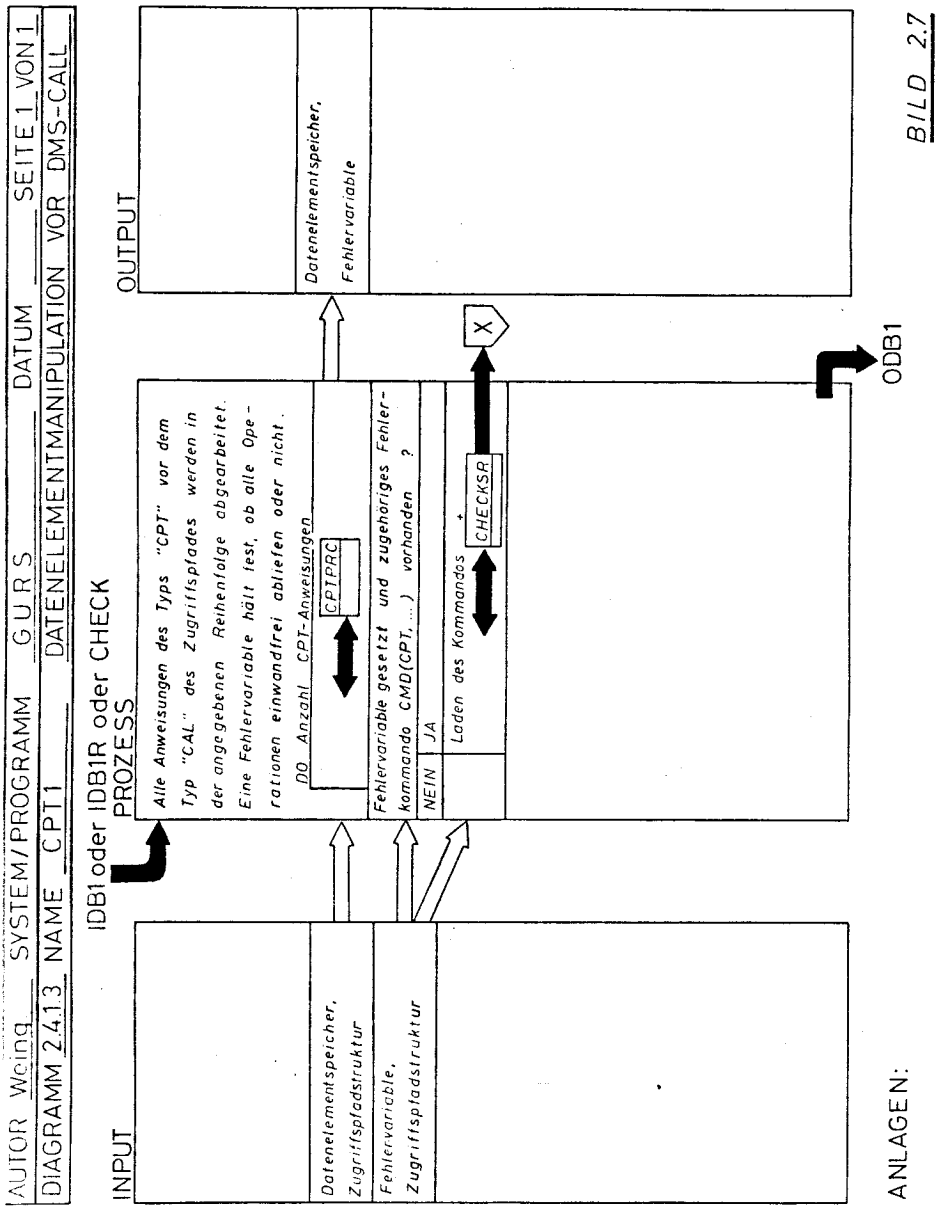


BILD 2.7

AUTOR Weing.	SYSTEM/PROGRAMM GURS	DATUM	SEITE 1 VON 1
DIAGRAMM 2.4.1.4	NAME ODB1	DATENELEMENT-SPEICHERAUSZUG VOR DMS-CALL	

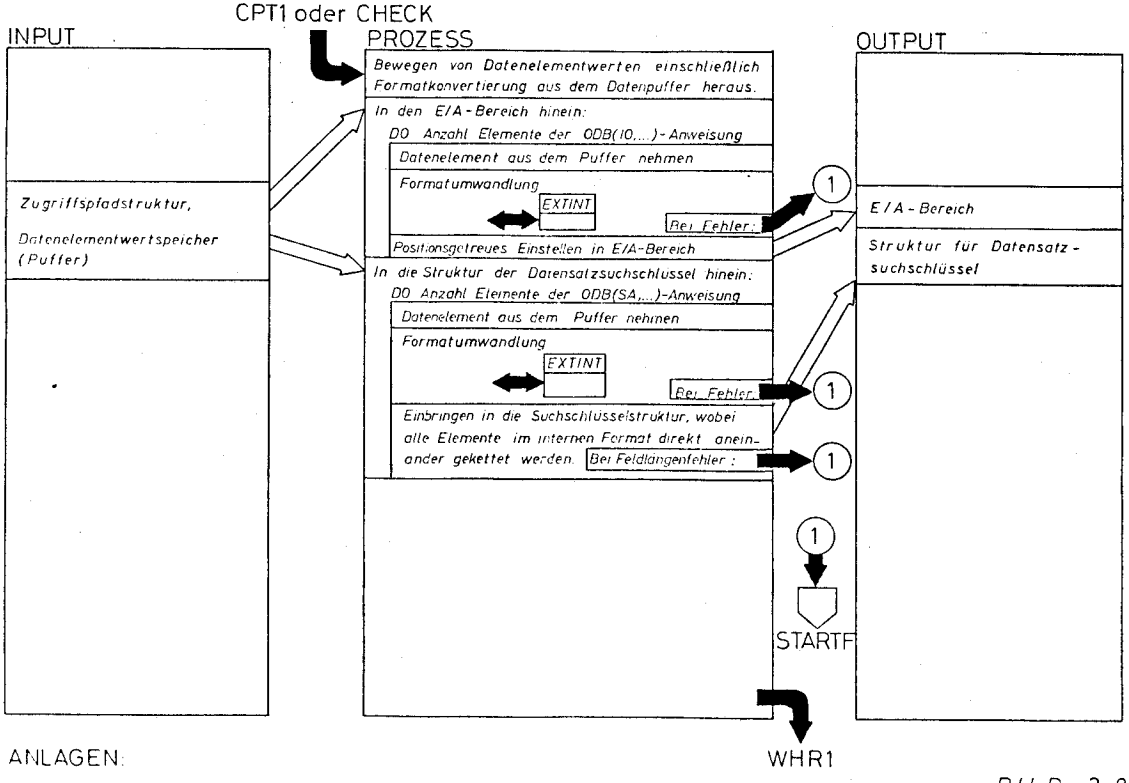


BILD 2.8

AUTOR Weing.	SYSTEM/PROGRAMM GURS	DATUM	SEITE 1 VON 1
DIAGRAMM 2.4.1.5	NAME WHR1	PRUFUNG LOG. BEDINGUNGEN VOR DMS-CALL	

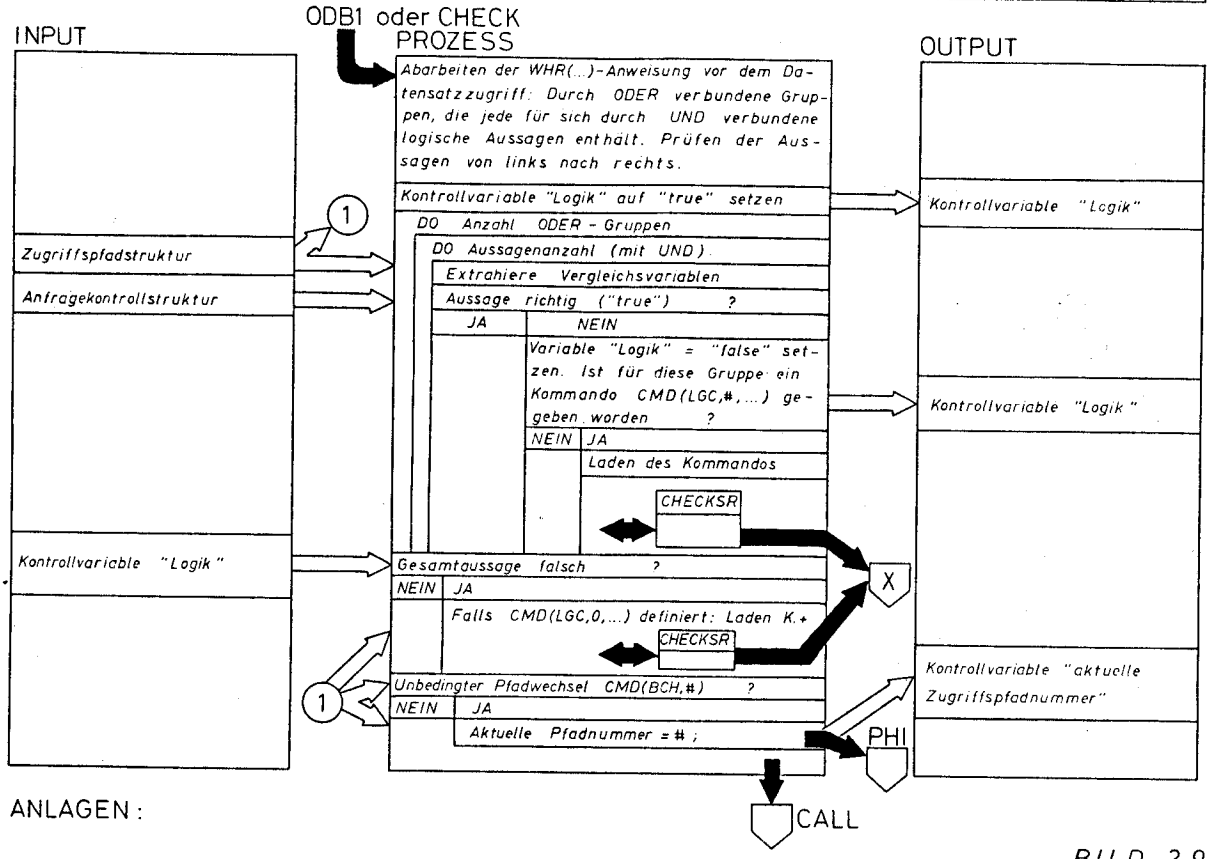


BILD 2.9

Die Art der Gliederung des Hauptprogramms in Programmblöcke ist prinzipiell diese:

Jeder Programmblock besitzt nur einen Eingang und einen normalen Ausgang, der auf den Eingang eines Programmblockes verweist. Darüber hinaus darf ein Programmblock wenige (bis drei) Neben- oder Fehlerausgänge besitzen. Verzweigungen zu Blockeingängen werden grundsätzlich über den Blocknamen programmiert: Makros des Vorübersetzers sorgen für die richtige GOTO-Auflösung. Die Verbindung der Programmblöcke kann über eine Dialogeingabeunterbrechung oder ohne diese direkt erfolgen. Der Unterschied wird durch zwei Typen von Blockendeanweisungen generiert.

Eine Detaildokumentation der Programmblöcke war durch den geringen Codeumfang und vorangestellten und eingestreuten Kommentar in den meisten Fällen nicht nötig. Als Beispiel soll der Block CPT1 dienen, Bild 2.10.

```
/*********************************************************/
/*      SPRACHAUSDRUCKVERARBEITUNG VOR CAL()             */
/*      CPT(...)                                         */
/*      DIE DEFINIERTEN DATENELEMENTVERARBEITUNGEN       */
/*      WERDEN DURCHGEFUEHRT. DABEI WIRD UEBER EINE      */
/*      FEHLERVARIABLE DER AUSGANG UEBER CMD(CPT,...)    */
/*      AM ENDE DER VERARBEITUNGSSCHLEIFE AKTIVIERT.     */
/*********************************************************/

/****************GURS***BLOCK***DIAGRAMNAME=CPT1**************/
DCL CPT1 BIN FIXED INIT(      12); ITZSUB=CPT1; NEXT(      12):
  /*****************************************************************/;
/*****COMPUTES BEFORE CALL**MAKROGESTEUERT****/
```

```
DO I=PTH(APTHNR).CPTSTRT(1) TO PTH(APTHNR).CPTANZ(1)  +
             PTH(APTHNR).CPTSTRT(1) - 1;
ERG=0;
CALL CPTPRC(PTH,APTHNR,QCTL,QDCTL,PAGE,WORKBUF,I,
        BIT( ODDCL( PTH(APTHNR).CPTNRL(I) ).AFG),
        BIT( ODDCL( PTH(APTHNR).CPTNRL(I) ).AFK),  ERG);
IF ERG>0  & PTH(APTHNR).CMB(4)<0 THEN PTH(APTHNR).CMB(4)=
                  PTH(APTHNR).CMB(4) * -1;
END;
```

```
/***CMD(CPT,...)***FEHLER BEI CPT, COMMAND-VERARBEITG***/
IF PTH(APTHNR).CMB(4)>0 THEN DO;
PAGE=CMDTXT( PTH(APTHNR).CMB(4) );
PTH(APTHNR).CMB(4) = -1*PTH(APTHNR).CMB(4); /**RUECKSETZEN**/
ERG=0;
CALL CHECKSR;
END;
```

```
/*****BLOCKENDE*****/ NEXSUB=ODB1; GOTO OLD;
```

BILD 2.10

Das interpretierende System und der Übersetzer sind in dem System GURS (Generelles Update- und Retrievalsystem) enthalten, das sowohl am Bildschirm, als auch als Stapelversion in seinen Funktionen identisch zur Verfügung stehen.

Die Programmiersprache ist PL/1. Der Kompiler (41) unterstützt die strukturierte Programmierung. Codegenerierung für die Ablaufsteuerung der Moduln wurde mit dem Vorübersetzer durchgeführt. Um den Codeaufwand (Hauptspeichergröße) zu beschränken, wurde mit Hilfe von Vorübersetz*ung*, auf die hier nicht näher eingegangen werden soll, die Möglichkeit eingerichtet, Funktionen und DMS-Schnittstellen variabel zusammenzugenerieren. Diese Möglichkeit war sinnvoll, da der Speicherplatz für ein Programm (batch partition) des Betriebssystems (OS-MFT) im MSH auf 300K Byte begrenzt wurde und nur für Sonderaktionen vergrößert zur Verfügung stand. Der Nachteil war eine erforderliche Versionsplanung und der Versionseinsatz je nach Aufgabe. Bei Ausschluß z.B. der relationalen Abfragemöglichkeit und der Schnittstelle PATSY (Seite 55) ist die sonst vollständige Version im Stapelbetrieb lauffähig. Im Dialogbetrieb steht genügend Platz zur Verfügung, so daß hier mit der Maximalversion gearbeitet wird. Der Platzbedarf ist hier mit 400K verglichen mit dem Patientenaufnahmeprogramm mit mehr als 400K in vertretbaren Grenzen. Auch das Patientenauskunftssystem PIDS hat trotz Einsatz von Überlagerungstechnik (overlay) noch 350KByte Platzerfordernis.

2.3.2.2 Ablaufgerüst

Programmblöcke (Moduln) werden in einen Ablaufrahmen eingefügt, der der Programmteilbildung des TIMS (Teleprocessing Interface Makro System = Fernverarbeitungsmonitor des MSH) entspricht. Bei TIMS muß der Anwender für die Gliederung seines Programms folgende reservierte Variablen benutzen:

NEXT(N) LABEL	*(Programmarkenfelder für die Blockeingänge)*
NEXSUB BIN FIXED	*(Verzweigungsindex, Verweis auf Programmarke)*
GOTO MES	*(Anweisung für das Einlesen einer Dialogeingabe (Terminaleingabe))*
PAGE CHAR(*) VAR	*(Variabel lange Eingabevariable für Nachrichten)*
GOTO OLD	*(Blockverzweigung ohne Dialogeingabe)*
CHECKSR: PROC	*(internes, selbst zu kodierendes Eingabeprüfunterprogramm, im folgenden mit CHECK bezeichnet).*

Das Makro Interface System TIMS bietet weitere Möglichkeiten, auf die nicht eingegangen werden soll (68).

Die Blockbildung in einem TIMS Anwendungsprogramm wird in Bild 2.11 verdeutlicht (Source = Quellcode):

```
PROGRAMM:   PROCEDURE  OPTIONS (MAIN);
% INCLUDE   MACLIB (TIMS);
/* das über Vorübersetzer eingeholte Teleprocessing
   Interface Makro System enthält u.a. folgende
   Programmarken:
   MES:      - einholen einer Terminaleingabe -
             CALL CHECK     - Prüfprozedur -
   OLD:      GO TO  NEXT (NEXSUB)                 */
/* das eigene Programm beginnt danach:             */
NEXT (1):          /* z.B. Eingabeaufforderung     */
- Source -
NEXSUB = 2;
GO TO MES;
NEXT(2):           /* z.B. Eingabe verarbeiten     */
-Source-
NEXSUB = 1;
GO TO OLD;
CHECK:  PROCEDURE;
- Source -
END CHECK;
END PROGRAMM;
```

BILD 2.11

Programmablaufgerüst für interaktive PL/1-Programme im Medizinischen System Hannover

Diese Art der Blockbildung wurde der größeren Sicherheit und Standardisierung wegen modifiziert. Dazu wurde die Verwaltung der TIMS-Variablen NEXT(*), NEXSUB, MES und OLD über Vorübersetzer-Makros automatisiert.

Der Vorteil ist, daß Blockbildungen und Blockverzweigungen über mnemonische Namen statt über Zahlen programmierbar sind. Bei 25 verschiedenen Blöcken bedeutet dies eine spürbare Vereinfachung. Das Beispiel für TIMS programmiert sich jetzt auf diese Weise:

Aufruf für TIMS und eigene Makros

```
BLOCK(EINGABE);
...Quellcode...
EOBMES(RECHNEN);

BLOCK(RECHNEN);
...Quellcode...
EOBOLD(EINGABE);

CHECK: PROC;
...Quellcode...
END CHECK;
```

Der Blockanfang wird durch das Makro BLOCK generiert. Die unterschiedlichen Blockabschlüsse werden durch "End Of Block" EOBMES bzw. EOBOLD erzeugt. Nebenausgänge werden über das Makro SPRUNG realisiert: um den Programmteil Fehlerauflaufpunkt in das Beispielprogramm einzufügen, sind die folgenden geringen und übersichtlichen Ergänzungen nötig:

Aufruf für TIMS und eigene Makros

```
BLOCK(EINGABE);
...Quellcode...
EOBMES(RECHNEN);

BLOCK(FEHLER);
...Quellcode...
EOBMES(RECHNEN);

BLOCK(RECHNEN);
...Quellcode 1...
IF Fehler DO;
 SPRUNG(FEHLER);
END;
...Quellcode 2...
EOBOLD(EINGABE);
```

Unabhängig von der Blockbildung im Hauptprogramm selbst, das die Ablaufsteuerung enthält, erfolgte eine weitere Modularisierung durch interne und externe Unterprogramme.

2.3.2.3 Dialog-/Stapelversion

Das Unterprogramm CHECK wird nach jeder Eingabe vom Bildschirm durch das TIMS aufgerufen. Im Unterprogramm befindet sich die außerhalb TIMS zu programmierende Untersuchung der Eingabe auf das Enthalten-

sein von Kommandos.
Einige, die dem Setzen von Steuervariablen dienen (BRAKE/NOBRAKE, LIST/NOLIST, COUNT/NOCOUNT, etc.) oder das Wegschreiben einer DB-Suche für späteren Aufruf (QPUTnr) werden direkt durchgeführt. Die Steuerung geht danach an die aufrufende Programmstelle zurück. Andere Kommandos bewirken z.B. die Verzweigung zu bestimmten Programmblöcken (QUERY zu QUL, STOP zu ST_F, START zu QUE).
Das Unterprogramm CHECK ist der einzige Modul, der mehrere (über 20) Ausgänge zu Programmblöcken enthält. Dafür wird mit besonderer Sorgfalt kodiert.

Für die Stapelversion muß der TIMS-Anteil durch einen gegenüber dem Programmrest kompatiblen Teil ersetzt werden. Der "Dialog" ist durch Ein- und Ausgaben ersetzt. Diese Art des Dialogs ist eher als Kommandofolge zu verstehen, die zumeist in der interaktiven Version vorgetestet wurde. Der TIMS-Anteil der Stapelversion hat folgenden Aufbau:

```
"Deklaration der zu verwendenden TIMS-Variablen"
MES:  GET FILE(SYSIN) LIST (PAGE);
      CALL CHECK;
OLD:  GOTO NEXT(NEXSUB);

PUT:  PROC;
"Verarbeitung für FILE(SYSPRINT)-Ausgabe"
END PUT;
```

2.3.3 Syntaxanalyse und Übersetzer

Die Modulgliederung wurde in Bild 2.5 dargestellt. Hier soll das Verfahren der Syntaxanalyse und Übersetzung in Datenstrukturen, die den Programmablauf des Interpreters steuern, beschrieben werden. Als Beispiel dient die Verarbeitung der CPT-Anweisung. Zur Erläuterung wird das Bild 2.12 benötigt.

Die Tabelle der Anweisungstypen in Bild 2.12 spiegelt durch den Tabellenplatz die Stellung innerhalb der Verarbeitungssequenz der Pfadmaschine (wie in Bild 2.2 erläutert). So wird der Verarbeitungsauftrag abgesucht nach Anweisungstypen, wobei eine bestimmte Reihenfolge angenommen wird. Es können zuerst nur DDC-Typen auftreten, dann muß als erster nicht DDC-Typ der Typ SFS des ersten Zugriffspfades auftreten, usw. Nachdem so der ganze Verarbeitungsausdruck syntaktisch abgeprüft und die Stellung der Anweisungen zueinander registriert wurde, werden alle Klammerinhalte in eine den Interpreter steuernde Datenstruktur übersetzt. Diese sog. Pfadstruktur enthält je definiertem Pfad alle Entscheidungen, um bei Durchlauf der Programmblöcke die richtigen Aktionen zu aktivieren oder zu übergehen.

Die Anweisungstypen sind in der Tabelle (POSITION, TYP) entsprechend der Pfadmaschinenlogik definiert (Bild 2.12, unten). Ein Datenverarbeitungsauftrag möge wie in Bild 2.12, Spalte 1, dargestellt, aus

zwei Zugriffspfaden bestehen und im ersten Pfad vor dem Datensatzzugriff "CAL()" drei CPT-Anweisungen, danach eine und im zweiten Pfad nach "CAL()" ebenfalls eine CPT-Anweisung enthalten.

Als Ergebnis der Syntaxanalyse ist die Tabelle (I, J, POS1, POS2) durch pfadweises Abarbeiten des Verarbeitungsauftrages in PDPL-Notation entstanden. Dabei bezeichnen POS1 und POS2 die Klammerausdrücke der Verarbeitungsanweisungen, I die Pfadnummer (aus SFS-Anweisung) und J die Positionsnummer POSITION des Anweisungstyps in der Anweisungstabelle. Hier ist die Aufgabe des Platzhalters CAL() deutlich erkennbar, weil derselbe Anweisungstyp CPT davor und danach auftreten kann. Dieses Konzept der relativen Position zu Platzmarkierungen (SSA() und CAL()) ermöglicht eine Reduzierung der Zahl der Anweisungstypen bei gleichzeitiger Anschaulichkeit der Reihenfolge der ablaufenden Verarbeitung im Interpreter.

Bezüglich CPT enthält (I, J, POS1, POS2) als Ergebnis der Syntaxanalyse (Bild 2.12, Spalte 2) für den ersten Pfad die Quadrupel (1,13, p1,p2), (1,13,p3,p4), (1,13,p5,p6) und (1,21,p7,p8). Der zweite Zugriffspfad enthält bezüglich CPT lediglich: (2,21,p9,p10). Der Übersetzer hat jetzt die Aufgabe, die Klammerausdrücke des Auftrages mit POS1 und POS2 und der Bedeutung J (nach Anweisungstypenliste: was wann im Pfad durchzuführen ist) in Daten der Zugriffspfadstruktur zu übersetzen.

Für die CPT-Anweisungen ist der Pfadstruktur ein Hilfsspeicher (Computespeicher) zugeordnet, in den die durchzuführenden Operationen ausführungsgerecht eingetragen werden (Bild 2.12, Spalte 3). Deshalb besitzt die Pfadstruktur (Bild 2.12, letzte Spalte) nur Zeiger, die auf die entsprechenden Zeilen des Computespeichers verweisen. Die Zeiger des ersten Pfades heißen

PFAD(1).VONCPT1 und PFAD(1).ANZCPT1
für CPT-Anweisungen vor dem Datensatzzugriff (vor CAL)
PFAD(1).VONCPT2 und PFAD(1).ANZCPT2
für die CPT-Anweisungen nach dem Datensatzzugruff. VONCPT ist die Startadresse im Computespeicher, ANZCPT ist die Anzahl der zu verarbeitenden Zeilen des Computespeichers.

Der formale Aufbau einer CPT-Anweisung war

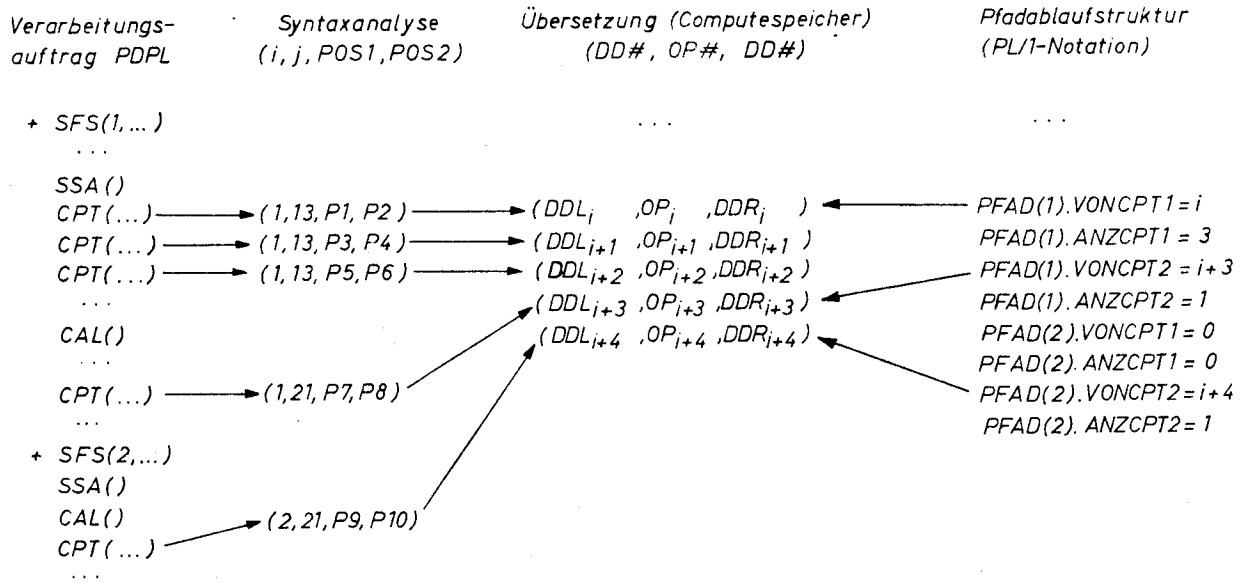
CPT (Datenelementnummer Operator Datenelementnummer)
oder
CPT (Datenelementnummer Operator Zeichenkette).

Der Computespeicher ist ein Abbild des Klammerinhaltes, allerdings nicht mehr mit noch zu interpretierenden Operatoren, sondern entsprechenden Steuerungszahlen für die unmittelbare Ausführung der Operationen. In Bild 2.12 enthält der Computespeicher (DDL, OP, DDR). DDL bezieht sich auf das Datenelement links vom Operator, DDR auf dasjenige rechts vom Operator bzw. auf eine Zeichenkettennummer eines Zeichenkettenspeichers. Der Index i ist die Numerierung der Zeilen des Computespeichers.

Der Interpreter im Programmblock CPT1 (Bild 2.10) enthält unter anderem die Anweisungen:

```
DO i = VONCPT1(APTH) TO VONCPT1(APTH) +ANZCPT1(APTH) - 1;
 ...
 CALL CPTPRC(...,
                Computespeicherzeile(i),
                Datenelementspeicher,
                Zeichenkettenspeicher,
                Fehlervariable);
 ...
END;
```

APTH ist die Nummer des jeweils aktiven Zugriffspfades. Die Anzahl der CPT-Anweisungen ANZCPT1(APTH) und der Start dieser Anweisungen im CPT-Speicher VONCPT1(APTH) war vor der Sprachübersetzung mit Null initialisiert, so daß die Verarbeitungsschleife ohne ordnungsgemäße Übersetzung einer entsprechenden Anweisung in diese Steuerungszahlen nicht aktiviert wird. Durch die so detaillierte Vorbereitung der Verarbeitung durch den Übersetzer ist der Interpreter bei der Durchführung nur mit einfachen und schnellen Ablaufentscheidungen befaßt. Vom Prinzip her werden auch die anderen Anweisungstypen ähnlich umgesetzt. Darauf soll im einzelnen nicht eingegangen werden.



· Anweisungstypenliste:

POSITION	1	2	3	4	5	6		10		13		17		21		26
TYP	NNN	DDC	NNN	SFS	NNN	ACC	...	SSA	...	CPT	...	CAL	...	CPT	...	NNN

BILD 2.12 Syntaxanalyse- und Übersetzerergebnisse, Erläuterungen im Text

2.3.4 Interpreter

Der Interpreter besteht aus einer fest vorgegebenen Sequenz von Programmblöcken (Bild 2.4 und 2.6), die teilweise unbedingt, teilweise bedingt miteinander verbunden werden. Der Verarbeitungsauftrag aktiviert oder deaktiviert die vorgegebenen Funktionen der jeweiligen Blöcke eines Pfades. Die Reihenfolge der Anweisungen ist also nicht frei wählbar. Diese Einschränkung gegenüber anderen Programmiersprachen vereinfacht die Programmierung des Gesamtsystems.

Die Bedeutung und Sequenz der Programmblöcke ist in Bild 2.13 mit kurzen Erklärungen wiedergegeben.

Folgende Bemerkung erscheint noch wichtig:
Die Sprachausdruckverarbeitung IDB1 ist angegeben (Seite 149) mit IDB(FI,...) und IDB(EX,...). Die anderen prinzipiellen Möglichkeiten IDB(IO,...) und IDB(KF,...) sind von der Syntaxdefinition und auch vom Übersetzer her vorgesehen, d.h. entsprechende Felder der Zugriffspfadstruktur werden gesetzt, aber vom Interpreter aus praktischen Gründen nicht genutzt: Es ist nicht der Regelfall, daß Datenelemente vor dem Datensatzzugriff aus dem Ein/Ausgabebereich (IO) extrahiert werden usw. Bei Bedarf ist es jedoch einfach, den Interpreter zu erweitern. Dies gilt auch für die Benutzung einer Sonderschnittstelle IDB(R...), die zum Teil für die relationale Sprachverarbeitung dem System selbst vorbehalten ist und hier nicht beschrieben wird.

2.3.5 Zusätzliche Stapelfunktionen

Neben der Stapelversion von GURS gibt es einige Hilfsfunktionen, auf die ebenfalls nicht weiter eingegangen wird:

- *Datensicherung für die Steuerdatei (Datenelemente, Hierarchiebeschreibungen, Aufträge, etc.)*
- *Einspielprogramm für Hierarchiedefinitionen*
- *Ausdrucken verfügbarer Datenelementbeschreibungen*
- *Ausdrucken der gespeicherten Datenverarbeitungsaufträge*
- *Überspielen von Datenelementbeschreibungen aus dem Daten- und Programmbeschreibungssystem DAPRO (73) des MSH*
- *Namensinvertierung der Datenelemente.*

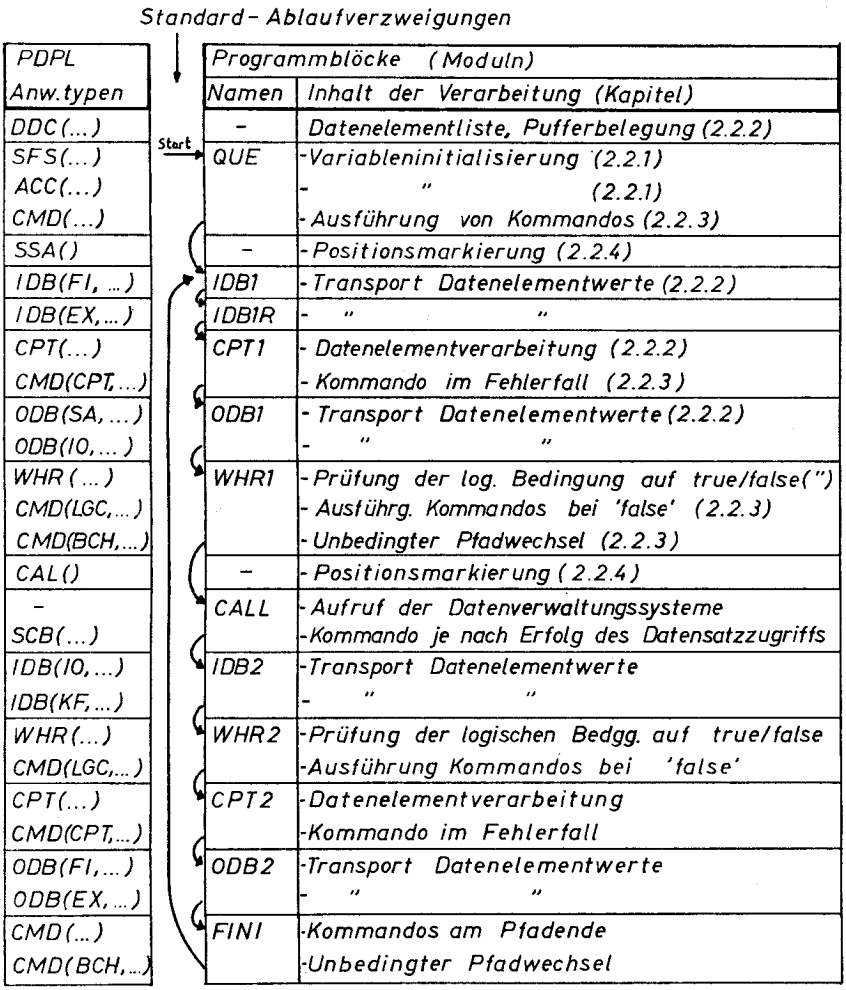


BILD 2.13

Darstellung aller Verarbeitungsanweisungen eines PDPL-Pfades (im linken Teil) und Gegenüberstellung der Moduln des Interpreters. Die Sequenz des Durchlaufs durch die Moduln und die Bildung einer pfadinternen Verarbeitungsschleife zwischen FINI und IDB1 wird verdeutlicht. Einzelheiten zu den Anweisungstypen sind neben den hier angegebenen in Kapitel 6 zu finden.

3. LEISTUNGSBEWERTUNG

3.1 Qualitative Bewertung

Das eine Anweisungsfolge in PDPL-Notation verarbeitende System, also der Sprachprozessor, ist das System GURS (Generelles Update- und Retrievalsystem), das sowohl am Bildschirm, als auch per Staperverarbeitung identisch zur Verfügung steht. Dadurch können Sprachausdrücke interaktiv getestet werden, jedoch im Stapelbetrieb endgültig ausgeführt werden. Sprachausdrücke lassen sich speichern und für die Ausführung abrufen.

Anwendungen zeigen, daß Probleme, die zu Schwachstellen (vergl. 1.2.2) des Krankenhausinformationssystems gehören, zum größten Teil mit der Einführung der PDPL lösbar geworden sind:

1. Die Handhabung der PDPL ist einfacher als das Erstellen von PL/1 und DL/1 Programmen. Durch eine Schnittstellentransformation über einen PDPL-Ausdruck z.B. konnte die "Patienten Entlassungsliste" des MSH mit der Komponente "Druck akkumulierter Laborergebnislisten" für einen Testbetrieb in kürzester Zeit verbunden werden, so daß bei entlassenen Patienten bestimmter Stationen akkumulierte Laborlisten für die Arztbrieferstellung gedruckt werden.
2. Ad hoc-Anfragen, wie z.B. "Welche Patienten mit Wahlleistung Arzt waren wie lange in 1977 auf Station xyz" wurden am Bildschirm ausgetestet und im Stapelbetrieb durchgeführt. Die Schwelle der gar nicht bearbeitbaren Benutzerwünsche ist damit im MSH nachweislich gesunken.
3. Datenunabhängigkeit ist weitgehend durch das Datenbeschreibungssystem gegeben.
4. Eine Sprachübersetzung von einer SEQUEL-ähnlichen Sprache in die PDPL und anschließende Ausführung ist in einer Testversion realisiert. Damit wird ein stark erhöhter Benutzerkomfort angestrebt, so daß eventuell Nichtprogrammierer Datenbankauskünfte definieren können.
5. Für den Vergleich der Leistungsfähigkeit des Systems mit dem Datenbank-Suchrechner der TU Braunschweig wurden einzelne Relationen erstellt, die der hierarchischen Datenstruktur entstammen. Diese Restrukturierung der Daten zeigt die Brauchbarkeit des Konzeptes für diesen Zweck.

Der Codeaufwand für Verarbeitungsaufträge beträgt schätzungsweise 1/10 oder noch weniger im Verhältnis zu einer PL/1-Lösung. Dies bedeutet von vorneherein eine erhebliche Einsparung für Ad-hoc-Anfragen. Die Testzeiten für PL/1-Programme sind bei Rückkehrzeiten von bis einem Tag für Stapelverarbeitung ebenfalls unvergleichbar größer als der Dialogtest mit anschließender Stapelausführung. Die PDPL bedeutet gegenüber der bisherigen Programmerstellung also einen beträchtlichen Gewinn an Benutzereffektivität. Diese wird bezahlt mit einer eventuell größeren Belastung der Betriebsmittel des Rechners, insbesondere der CPU. Ein Vergleich mit einfacher sequentieller Datenbanksuche brachte eine Antwortzeitverschlechterung von 10 Prozent gegenüber der PL/1-Version. Kompliziertere Programme wurden nicht verglichen.

3.2 Quantitative und vergleichende Bewertung

3.2.1 Methoden der Leistungsmessung

Die verschiedenen Methoden der Leistungsermittlung sind (92):

- *Analytisches Verfahren*
- *Systemsimulation*
- *Leistungsmessung*
- *Benchmarktest*
- *Schreibtischanalyse*
- *Methodenmix.*

3.2.1.1 Analytisches Modell

Analytische Verfahren werden bereits bei kleineren Konfigurationen (gemeint sind Rechner) kompliziert und sind für mittlere und große Systeme ungeeignet oder nicht realisierbar (21). Für die herausgelöste Betrachtung z.B. einer speziellen Zugriffsmethode, ist das analytische Verfahren jedoch einsetzbar (104). Wie noch gezeigt wird, ist allerdings im Falle einer IMS-Zugriffsmethode der unbekannte Zeitanteil des Datenverwaltungssystems (28, 81) ein Mehrfaches des ausrechenbaren Sekundärspeicherzugriffs, so daß durch Abschätzung des Verwaltungsanteils die mit der Methode prinzipiell erzielbare Genauigkeit wieder verloren geht.

3.2.1.2 Systemsimulation

Diese Methode ermöglicht grundsätzlich die Nachbildung von Systemen. Es wurde gezeigt, daß z.B. ein komplexes Informationssystem mit Datenbanken für ein Versicherungsunternehmen mit Erfolg simulierbar war (56). Dabei mußten Datenbankbeschreibungen, Arbeitslasten, Betriebssystemteile und das Hardwaresystem modelliert werden. Spezielle Simulationssprachen wie z.B. CSMP, DYNAMO, SIAS, SIMULA, GPSS oder PHASE II/FOREM sind verfügbar. Bei Simulationssystemen allerdings gerät man voll in die Problematik der Modellbildung (20, 27, 44), zu der es keine geschlossene Theorie gibt. So sind z.B. Fragen der Richtigkeit angenommener Analogien oder des Detaillierungsgrades nicht immer mit Sicherheit zu beantworten. Die Antwort der Datenbankforschung im Zusammenhang mit Simulation steht auf die beiden folgenden Grundfragen noch aus (9):

> *Welche allgemeine Erkenntnis vermitteln Aussagen, die durch Modellierung einer ganz konkreten Situation gewonnen wurden ?*
> *und*
> *Welchen Wert haben Erkenntnisse im konkreten Einzelfall, die aus allgemeiner Simulation entspringen ?*

Hill (32) erwähnt, daß ein Vielzweckmodell kaum für Fragestellungen, die zur Zeit der Modellentwicklung unbekannt waren, sensitiv ist.

Für die Simulation eines realen Systems werden teure Fachleute und teure Rechenzeit benötigt. Der Modellbildungsprozeß erfordert darüber

hinaus zur Modellvalidierung in jedem Fall Vergleichsmessungen. "Only by measurement do we really get smart" (32). Stimler (92) stellt fünf Vorteilen der Systemsimulation vier Nachteile gegenüber. Andere Autoren stehen der Simulation skeptisch gegenüber: Terplan (95) stellt fest, daß komplexe Systeme, besonders dann, wenn Betriebsmitteleigenschaften mit erfaßt werden müssen, kaum mit vertretbarem Aufwand zu modellieren sind. Blaser (9) vermutet, daß Simulationssysteme, wie z.B. FOREM (55, 83) zu schwach zu sein scheinen, um das gesamte Systemverhalten zu simulieren. Deshalb setze sich die Meinung durch, daß man speziell auf Datenbankfragen zugeschnittene Simulationssysteme benötige.

3.2.1.3 Leistungsmessung

Die Leistungsmessung geschieht prinzipiell mit Hardware- oder Softwaremonitoren (28, 31, 94, 95). Diese erlauben sehr genaue Aussagen über die prozentuale Auslastung von Betriebsmitteln, so daß Engpässe gut erkennbar werden. Auch lassen sich z.B. durch logische Und-Verknüpfungen überlappte Belegungszeiten ermitteln, so daß auch Aussagen zur Konfigurationsoptimierung ableitbar sind. Die Daten der Monitore sind repräsentativ für die gesamte Rechnerlast, Jobabrechnungsroutinen geben Anhalt über Betriebsmittelbelegung einzelner Verarbeitungssysteme. Die ermittelten Datenmengen von Monitoren sind sehr groß, so daß deren sinnvolle Verwendung ein Problem für sich ist.

3.2.1.4 Benchmark

Der Benchmarktest beinhaltet die über eine definierte Zeit durchgeführte Verarbeitung einer begrenzten Anzahl von Testprogrammen.

3.2.1.5 Schreibtischanalyse

Von Stimler (92) wird als die kostengünstigste und unverzichtbare Methode zur Leistungsermittlung die Schreibtischanalyse genannt. Sie ist analog der Überschlagsrechnung zu sehen. Sie umfaßt die Definition der Hardware, der Leistungsmerkmale der Systemkomponenten, des Arbeitsprofiles und die darauf beruhende vereinfachte Nachrechnung oder Abschätzung von z.B. Antwortzeitverhalten. Neben Simulation und Messung unterstützt sie die Modellvalidierung besonders gut. In einigen Fällen reicht sie alleine aus.

3.2.1.6 Vorgehensweise

Anhand der Literatur ergibt sich zur Leistungsermittlung eines Systems folgende sinnvolle Vorgehensweise:

1. Festlegung der Terminologie
2. Definition des Zieles der Leistungsermittlung
3. Systembeschreibung und angenommenes Systemmodell
4. Erstellung Leistungsmodell (performance model) zur Systemstruktur und zur angenommenen Arbeitslast

5. Validierung der Modellergebnisse durch Messung und Schreibtischanalyse
6. Ergebnisdokumentation

Unter Umständen ist ein wiederholtes Durchlaufen der Punkte 2 bis 5 erforderlich, bis das Modell zuverlässige Ergebnisse bringt (Modellvalidierung).

3.2.2 Motivation zur vergleichenden Leistungsbewertung

Auf der internationalen Konferenz "Very Large Data Bases" 1978 in Berlin wurde ein Leistungsvergleich zwischen einem sogenannten GPC (General Purpose Computer) und einem DBC (Data Base Computer) vorgestellt (6). In beiden Fällen lag die Annahme einer relationalen Datenverwaltung zugrunde. Das Ergebnis war ein im allgemeinen besseres Abschneiden des DBC-Konzeptes, wenn auch bezweifelt werden muß, daß die Annahme richtig ist, daß ein DBC-Konzept keinen Aufwand für die Optimierung der Anfragebearbeitung treiben müsse. Die Durchführung der Verarbeitung von Datenanfragen erfordert rechnerverarbeitbare Informationen und Modelle über die vorhandenen Datenstrukturen, da sonst z.B. unsinnige Anfragen, wie auch für die Antwortzeit völlig unbefriedigende Reihenfolgen von Mengenoperationen die Folge sind. Anstelle der Zeiger z.B. des GPC-Konzeptes sind wegen der normalisierten Form im GPC häufige Mehrfachspeicherung von Schlüsselfeldern für die Darstellung der bedeutungsmäßigen Abhängigkeiten der Felder erforderlich.

Da Schnelligkeit der erstellten Programme für eine Produktionsaufgabe ein spezielles Anliegen konventioneller Datenbanksysteme ist, kann es sein, daß das DBC-Konzept mit einem relationalen Konzept zwar gegenüber einem gleichen relationalen Konzept im GPC besser ist, jedoch gegenüber einem hierarchischen und für Anwendungen spezialisierten GPC-Konzept dennoch unterlegen ist.

Da der Suchrechner (SURE) der Arbeitsgruppe des Instituts für DV-Anlagen und des Lehrstuhls D für Informatik in Braunschweig ein DBC-Konzept, die PDPL ein GPC-Konzept ist und beide für den Anschluß relationaler Datenanfragen vorgesehen sind, ist ein Antwortzeitvergleich sinnvoll und in Relation zur erwähnten Untersuchung auch reizvoll.

3.2.3 Suchrechnerkonzept

Durch die Verlagerung von Intelligenz in die Speichermedien sollen Suchvorgänge und Verknüpfungsfunktionen von Datenbanksystemen von der CPU und dem Arbeitsspeicher weg in die Peripherie verlagert werden. Ein kleiner Steuerrechner bildet mit dieser Spezialperipherie den Datenbankrechner, Bild 3.1. Das Konzept wurde detailliert dargestellt (46, 53, 54, 89, 90, 105), so daß hier nur eine Zusammenfassung erfolgen soll.

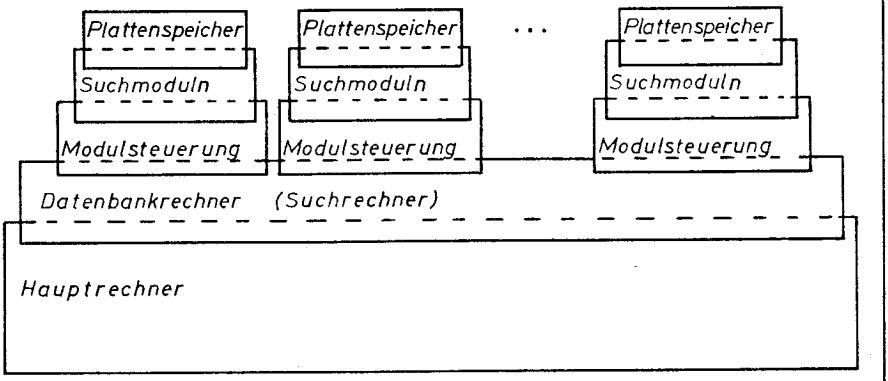


BILD 3.1

Darstellung des Suchrechnerkonzeptes als an einen Großrechner koppelbaren Datenbankrechner mit Spezialelektronik

Die speziell entwickelte Hardware durchsucht den gesamten Datenbestand. Jeder an der Platte gelesene Satz wird ohne Speicherung mit dem Lesevorgang schritthaltend mit einer Vielzahl von Suchanfragen verglichen. Treffer werden in Treffervektoren als Satzadressen abgelegt. Die Lesegeschwindigkeit der Sätze wird gegenüber herkömmlichen Geräten dadurch erhöht, daß von allen Spuren eines Zylinders verschachtelt gleichzeitig gelesen wird.

Die Struktur der Sätze wird durch Sonderzeichen selbstbeschreibend aufgebaut. Damit entfallen Indexlisten oder andere Adressverweise vollständig, der Nebeneffekt ist eine Reduzierung des Plattenbedarfs. Für Binärzeichen steht ein Umschaltsonderzeichen zur Verfügung.

Die Suchanfragen werden in Suchmoduln genannten Hardwareeinheiten am Plattenspeicher gespeichert. Die Sätze des Datenbestandes werden nacheinander Byte für Byte den Suchmoduln zugeführt. Die Sonderzeichen zeigen den Suchmoduln die Unterteilung des Datenstromes in Felder, Kategorien und Sätze an. Die Prüfung der Suchanfrage geschieht feldweise durch einen Satz von Vergleichsbefehlen und Befehlen zur logischen Zusammenführung von Teilergebnissen einer Anfrage.

Suchanfragen wie die folgenden sind möglich:

```
Suche einen Datensatz, für den gilt
es gibt  (Vorname = 'Heinrich') und (Nachname = 'Mueller')
         und ((Wohnort = 'Bonn') oder (Gehalt < 2500)).
```

Die Suchtechnik kann im weitesten Sinn als assoziativ (inhaltsadressiert) bezeichnet werden.

Realisiert ist die Hardware, die Suchmodulebene und ein Simulationspaket. Ein relationales Datenverwaltungssystem wird angestrebt, ist aber noch nicht realisiert.
Die Syntax des Suchmodul-Assemblers ist im Kapitel 6 beigefügt.

3.
2.4 Durchführung der Leistungsermittlung

Das Vorgehen entspricht den in 3.2.1.6 angegebenen Schritten.

1. Die Terminologie ist im Rahmen der Arbeit bereits genügend definiert. Bei Bedarf erfolgen weitere Erläuterungen an den Stellen, an denen neue Begriffe auftreten.

2. Ziel des Leistungsvergleiches soll die Ermittlung von Antwortzeitverhalten bei Datenbankanfragen in zwei unterschiedlichen Systemen sein: Datenbanksuche mit Hilfe einer Sprachimplementierung in einem Mehrzweckrechner (GPC) und Datenbanksuchen mit Hilfe eines Datenbankrechners (DBC). Dazu sollen Datenanfragen aus dem KIS der Medizinischen Hochschule und die zugehörigen Datenbestände ausgewählt und die Antwortzeiten beider Systeme verglichen werden.

3. Die Systembeschreibung und die Systemmodellierung werden durch die Angabe von analytischen Antwortzeitmodellen für den Suchrechner und die in den Arbeitslasten verwendeten Datenorganisationsformen des MSH speziell angegeben.

   Man muß Einschränkungen definieren, unter denen ein Modell gültig ist und damit Vergröberungen in Kauf nehmen:
   Datenbanken sollen sich im reorganisierten Zustand befinden. Die HDAM (3.4.1.5) Datenbanken wurden aus HISAM (3.2.4.1.4) Datenbanken erzeugt. Dabei entstanden spezielle Speicherungseffekte, auf die eingegangen werden muß. Datensätze können auf mehrere Stücke (extents) der Platte aufgeteilt sein. Zwischen Primärbereich und Überlaufbereich, falls vorhanden, können anderweitig belegte Zylinder liegen, wodurch sich die Kammbewegung zwischen diesen Bereichen entsprechend erhöht. Primär- und Sekundärbereich können aber auch auf verschiedenen Platten liegen und damit eventuell auf verschiedenen Kanälen, was die Zugriffszeiten wiederum relativiert.

   Wenn eine Platte von mehreren Datenbeständen belegt ist, dann ist bei realem Betrieb das Antwortzeitverhalten stark abhängig von anderen Programmaktivitäten auf dieser Platte: statt der halben Umdrehungszeit für die Positionierung in der Spur wird die Zylindersuchzeit gemessen. Deshalb ist jede Betrachtung nur für die angegebenen Last- und Speicherbedingungen gültig.

   Alle Zeitmessungen werden im "realen System" durchgeführt, so daß die Seitenwechselzeit der virtuellen Adressierung (84, 96), die das analytische Modell nicht enthält, sich nicht auf die Antwortzeit auswirkt.
   Die reinen Suchzeiten auf den Speichermedien sind wegen entsprechender Blockung manchmal nur 1/5 der Verwaltungsarbeit des Datenverwaltungssystems im GPC, so daß Feinheiten an der Platteneinheit (23, 60, 102) oder bestimmte Scan-Verfahren (60) nicht mehr zu Buche schlagen. Die Zugriffszeiten setzen sich zusammen aus Zylindersuchzeiten, Positionierungszeiten in den Spuren, Daten-

übertragungszeiten und DMS-Verwaltungszeiten. Alle anderen Zeiten sind vernachlässigbar.

4. Ein realistisches Spektrum von Arbeitslasten wird definiert, die Datenstrukturen beider Systeme beschrieben und Antwortzeiten sowie deren Verhältnis als GPC:DBC-Faktor (6) bestimmt.

   Die Arbeitslast wird in Form einer SEQUEL-Anfrage definiert und in die PDPL bzw. Suchmodulsprache abgebildet, so daß die Effektivität der Sprachübersetzung nicht zu betrachten ist, zumal in beiden Systemen noch kein optimierter Übersetzer vorhanden ist und auch die Forschung hierzu noch keine allgemein gültigen Aussagen gemacht hat. Die Arbeitslast wird den folgenden Systemteilen des MSH entnommen:

   Patientenauskunftssystem
   Patientenauswertungssystem PATWERT
   Entlassungslistenprogramm
   Akkumulierte Labordatenberichte
   Einige "Ad hoc"-Auswertungen.

5. Die Modellvalidierung erfolgt unabhängig von den Arbeitslasten für die Zugriffsmethoden durch Zeitmessungen. Für den Suchrechner wurden keine Zeitmessungen durchgeführt, da hier die Voraussage der Zeit in drei abschätzbare Anteile zerfällt:

   1. Ermittlung der Trefferadressen durch gezielte Suchumdrehungen
   2. Konventionelles Lesen der Treffersätze (sequentiell oder gestreut)
   3. Eventuell weitere Treffersatzverarbeitung, wie z.B. Sortierung.

3.2.4.1 Antwortzeitmodelle und ihre Validierung

Als Speichereinheit sollen Platteneinheiten der Charakteristik der IBM 3330 und IBM 2314 angenommen werden, da diese im MSH Verwendung finden und die Kontrollmessungen daran vorgenommen wurden. Die Leistungsfähigkeit des Suchrechners soll wegen der Vergleichbarkeit auch auf IBM 3330 Einheiten bezogen werden. Folgende Größen beschreiben die Speichereinheitscharakteristik:

VARIAB.	BEDEUTUNG	3330	2314
ABS	Anzahl Byte pro Spur	13030	7294
ASZ	Anzahl Spuren pro Zylinder	19	20
AZP	Anzahl Zylinder pro Platte	808	200
HUP	0.5 Plattenumdrehungszeit	8,35ms	12,5ms
ZFZ	Zylinderfolgezeit	10ms	25ms
ZSZ	Zylindersuchzeit	40ms	87,5ms
ZMZ	Maximale Kammbewegungszeit	55ms	135ms
AW1	Anfangswert, Näherung 1	10ms	25ms
ST1	Steigung, Näherung 1	0,325	1,6ms/Zyl.
AW2	Anfangswert, Näherung 2	17,5ms	45ms
ST2	Steigung, Näherung 2	0,094	0,45ms/Zyl.
SBK	Spurblockkapazität	X	XX
BMS	Übertragungsgeschwindigkeit	806	312 Byte/ms
DBL	Stufe der Hierarchie eines Segmentes (Level)		
SGZ	Anzahl Segmenttypen nach Datenbankdefinition DBD		
ZIM	mittlere IMS Verwaltungszeit pro Segment (7ms)		
IMSFAK	Erfahrungswert als Faktor für die Korrektur von ZIM		
	IMSFAK =(0,9 + (DBL-1)/10 + 0,7•SGZ•DBL/100)		

TABELLE 3.1

Gemeinsame Variablen in allen Zugriffsmodellen und ihre Bedeutung.

Die Variablen ZFZ, ZSZ, SBK, ZIM, AW1, AW2, ST1 und ST2 werden im folgenden näher erläutert.

Die Zylinderfolgezeit ZFZ ist die mittlere Zeit, die für das Einstellen des Lesekamms auf den unmittelbar nächsten Zylinder (sequentieller Zugriff) benötigt wird. Die Zylindersuchzeit ist die mittlere Einstellzeit des Kammes auf einen beliebigen Zylinder der Platte (wahlfreier Zugriff).

Die Anzahl physischer Sätze (SBK = Spurblockkapazität) wird von IBM für die Speicherplatten wie folgt angegeben:

X 3330: SBK = (13165/(135+C+KL+DL)), mit KL=0 folgt C=0 und KL ≠ 0 folgt C=45

XX2314: SBK = 1 + (7294-B)/(101+C+1,0435 • (KL+DL)), mit KL=0 folgt C=0 und KL ≠ 0 folgt C=56

KL = Schlüssellänge

DL = Datenlänge

Die Größe ZIM=7ms entstammt einem GMD-Erfahrungsbericht (25). IMSFAK ist ein Erfahrungswert, der die Korrektur von ZIM in Abhängigkeit der Kompliziertheit der Verwaltungsarbeit von IMS ermöglichen soll. IMSFAK ist ein Algorithmus, der von der Anzahl der Segmenttypen in der Datenbank und von der hierarchischen Stufe des Zielsegmentes abhängt und in der in Tabelle 3.1 angegebenen Form als beste Lösung, bezogen auf die durchgeführten Zeitmessungen, empirisch ermittelt wurde. Der Faktor 0,7 führt bei der physischen Speicherungsform HIDAM (3.2.4.1.6) auf Fehler bei der Datenbank CONPAT (Bild 3.7) zwischen 2,5 und 15 %. Wenn für HIDAM statt des Faktors 0,7 ein Faktor 1,0 in der Gleichung angesetzt wäre, lägen die Abweichungen in der Datenbank CONPAT nur noch

zwischen 2,3 und 6,4 %. Es wurde jedoch aus Gründen der Einfachheit nicht noch eine weitere Abhängigkeit für IMSFAK von der Zugriffsfunktion oder Organisationsform eingebracht. Wie allgemein dieser Korrekturalgorithmus ist, kann nicht beurteilt werden.

Bei realistischen Dateiverteilungen kommt es vor, daß die Zylindersuchzeit nicht über die ganze Platte, sondern nur über der Datei betrachtet, anzugeben ist. Der Arm soll in diesem Fall nur im Bereich der Datei verstellt werden. Nach Wedekind (98) läßt sich keine exakte analytische Beziehung für die Zugriffsbewegungszeit in diesem Fall angeben. Es wird allerdings vorgeschlagen, zwei unterschiedliche Näherungsgeraden für die Berechnung der Zylinderüberquerungs- und Einstellzeit anzunehmen:

Zeit für N Zylinder: STi•N + AWi
i=1 für 0 < N < AZP/10 oder
i=2 für AZP/10 < N < AZP

Wenn eine Datei auf AZD Zylinder verteilt ist, bedeutet das nach Wedekind (98) eine mittlere Zylinderüberquerungszeit von t = Sti•N + AWi mit N=AZD/3.

Zur vergleichenden Leistungsbewertung müssen äquivalente Datenbestände und Arbeitslasten vorhanden sein. Wie hierarchische Strukturen in Relationen zu überführen sind, wurde z.B. in (98) beschrieben, Bild 3.2.

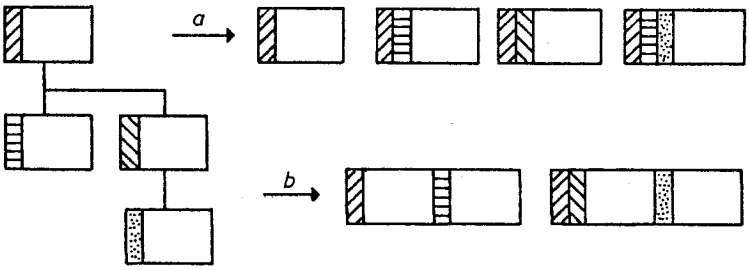


BILD 3.2

Entstehung ebener Dateien aus einer Hierarchie
a: Hinzufügen der Schlüssel übergeordneter Datensätze
b: Reduzieren der Dateianzahl durch Datenredundanz

Das einfachste Verfahren ist, jeden hierarchischen Satz mit dem Schlüssel der übergeordneten Sätze zu verbinden und für jeden Satztyp eine Relation anzulegen. Dieses mechanische Vorgehen wird dem Suchrechner nicht gerecht. Durch die Fähigkeit der Kategorienbildung mit Attributrelationen können zweistufige Hierarchien unter Umständen direkt abgebildet werden. Die Begrenzung liegt in der maximalen Satzlänge = Spurlänge. Wenn der Suchrechner seine Fähigkeiten voll ausschöpfen soll, wird ein suchrechnergerechtes Design nötig sein. Auch schon die Speicherung der Datensätze in durch die Schlüssel vorberechenbaren Bereichen (cluster) (6) erhöht die Schnelligkeit der Verarbeitung.

Eine teilweise automatische Umsetzung, wie sie hier erfolgt, ist nicht optimal, aber einfach realisierbar. Da eine Neustrukturierung zur Einschränkung der Relationenanzahl und das Auswählen von für die Arbeitslasten relevanter Felder im externen Format durchgeführt wurde, ergeben sich weitere Kritikmöglichkeiten. Aus Datenschutzgründen müßten die Daten anonymisiert werden.

Neben der reinen Plattenzugriffszeit ist noch die Zeit für die Ausführung eines Pfades durch den Interpreter abzuschätzen. Die Zeit ist abhängig von der Zahl und der Art der durchlaufenen Anweisungen. Bei den einfachen Zeitmessungen erfordert der Durchlauf eines Pfades:

38 IF-Abfragen, 49 Wertzuweisungen, 6 SUBSTR-Funktionen, 3 einfachste Unterprogrammaufrufe und ein einfaches (200 Codezeilen) Unterprogramm.

In diesem Beispiel ergibt sich eine Zeit von etwa 1,2ms. Der Aufwand des Betriebssystems wird über einen Korrekturfaktor eingebracht. Bei IMS zeigt sich, daß der Aufwand in der DMS-Schnittstelle die reinen Plattenzugriffszeiten um ein Mehrfaches übersteigt. Der Zeitbedarf für den Interpreter bei Umschaltung zwischen zwei Pfaden wurde zu 2,3ms ermittelt. Dieser Aufwand geht in die Zeitmodelle nicht ein, da dort im Wesentlichen für die Zeitmessungen nur ein Pfad gebraucht wurde. Je nach PDPL-Anweisungen ist in der Größenordnung eine Millisekunde als Interpreterzeit pro PDPL-Pfad (sog. Pfadzeit) anzunehmen.

.2.4.1.1 SAM (sequential access method)

SAM steht für Sequentielle Zugriffsmethode mit der Möglichkeit der Blockung von Datensätzen. Für das Zugriffszeitmodell werden neben den allgemein in 3.2.4.1 definierten Variablen noch folgende benötigt:

VARIABLE	I	BEDEUTUNG
PSL	I	Blocklänge in Byte
LSL	I	Datensatzlänge in Byte
BFAK	I	Blockungsfaktor PSL/LSL
ALS	I	Anzahl Datensätze in der Datei
APS	I	Anzahl Blöcke in der Datei APS=ALS/BFAK
ASD	I	Anzahl Spuren der Datei APS/SBK
AZD	I	Anzahl Zylinder der Datei AZD=ASD/ASZ
SBK	I	Spurblockkapazität

TABELLE 3.2

Zusätzliche Variablen für das Zugriffsmodell SAM und DAM.

Die Werte ASD und AZD sind aufzurunden. Betrachtet wird das Lesen der gesamten Datei mit der Funktion GET NEXT:

1. *Aufsetzen Startzylinder (ZSZ)*
2. *Positionierungen in der Spur (HUPxAPS)*
3. *Lesezeit (APSxPSL/BMS)*
4. *Zylinderüberquerungen (ZFZx(AZD-1)*
5. *Pfadzeit (GURS) ALSx1ms*
6. *Zeitkorrektur KORRFAKxAPS*

Mit der Zeit für die gesamte Datei kann der einzelne Zugriff auf einen Datensatz mit dem (ALS)-ten Teil abgeschätzt werden. Diese Zeit sollte mit der über die in GURS vorhandene Möglichkeit der Zeitmessung (CMD (TIME), Seite *143*) ermittelten Zeit übereinstimmen. Damit der Einfluß der Zeitbestimmung selbst (Systemaufruf und Ausgabe auf Lochkarte) nicht zu groß ist, wurde die Zeit grundsätzlich für 2000 DMS-Aufrufe genommen, insgesamt wurden nach Möglichkeit 8000 Zugriffe verwertet.

Beispiel für die Steuerdatei für GURS:

APS=ALS=8000, BFAK=1, Platte=3330, PSL=LSL=800, ASD=572,
AZD=31 SBK=14
Zeit für Datei=40+66800+7940+310+8000=83090ms
gemessen: 12x8000=96000ms, Fehler 13 %.

Eine Zeitkorrektur ist mit KORRFAK=1,6ms sinnvoll.

Diese Zeit steht für die in der Zeitaufstellung nicht enthaltenen, weil unbekannten Zeitanteile z.B. des OS-Betriebssystems für die gewählte Zugriffsmethode.

3.2.4.1.2 DAM (direct access method)

DAM steht für die Direktzugrifforganisation "Regional(1)" (39). Die benötigten Variablen entsprechen denen der Zugriffsmethode SAM. Unter der Annahme, daß ein Zufallszugriff auf einen Datensatz erfolgt und der Kamm irgendwo im Bereich der Datei (Datenbank) steht, ergibt sich folgende Rechnung:

1. *Positionieren auf Zylinder (AZDxSTi/3+AWi)*
2. *Positionierung in der Spur (HUP)*
3. *Lesezeit (PSL/BMS)*
4. *Pfadzeit 1ms*
5. *Zeitkorrektur KORRFAK 1,6ms*

Für die Steuerdatei GURS ergibt sich:

ALS=8000, PSL=LSL=800, BFAK=1, APS=8000, ASD=572, AZD=31
Zugriffszeit=31x0,325/3+10+8,35+1,0+1+1,6=25,3ms

Falls wahlfreier Zugriff (random access) über die ganze Platte erfolgt, so steigt die Zugriffszeit auf 52ms. Hieran ist besonders deutlich zu sehen, wie schädlich gleichzeitige Aktivität anderer Programme auf derselben Platteneinheit ist.

3.2.4.1.3 PATSY (patient summary system)

PATSY steht für Patientenzentraldatei (76) und umfaßt Patientenzentraldaten für das MSH für die Verifikation neuer und bereits vergebener Patientenidentifikationszahlen und das zugehörige als Unterprogramm aufrufbare Dateipflegeprogramm. Eine Namensinvertierung ist eingebunden. Die Organisationsform bedient sich einer Hashtechnik, die durch eine aus der tatsächlichen Schlüsselverteilung abgeleitete Tabelle eine gewünschte Gleichverteilung und Überlaufrate garantiert. Die Speicherplatzadresse zu einem Patientensatz weist auf einen "Bucket" genannten Block, der bis zu 35 selbst verwaltete Patientendatensätze aufnehmen kann. Da variabel lange Patientennamen ungekürzt untergebracht werden sollen, ist es Patientensätzen erlaubt, einen Textüberlaufsatz in unmittelbarer physischer Folge zu bilden (5 % = 0,85 mal je Bucket). Die Patientensätze liegen in sortierter Izahlfolge hintereinander (Geburtsdatum). Jeder Bucket kann seinerseits nur einen Überlaufbucket mit Platz für bis 35 weitere Patientensätze erzeugen. Eine regelmäßige Belegungsstatistik protokolliert Belegungen, so daß notfalls durch die Abfolge Entladen (unload), Berechnungstabelle für die Speicherplatzberechnung neu erzeugen und Rückspeichern (reload) der Daten wieder eine physische Gleichverteilung erzwungen wird. Primär- und Überlaufbereich sind in derselben Datei aneinandergrenzend definiert. Deshalb spielen die noch nicht belegten Überlaufbuckets in den Zeitbetrachtungen keine Rolle. Die Variablen für das Antwortzeitmodell sind:

VARIABLE	I	BEDEUTUNG
APS	I	Anzahl belegter Buckets einschließlich Überlauf
	I	(APS=(ALS/BELG))
BELG	I	mittlere Anzahl Patientensätze pro Bucket
BFAK	I	max Zahl Patientensätze im Bucket ohne Textsätze
PSL	I	Bucketlänge in Byte
LSL	I	Patientensatzlänge in Byte
ALS	I	Anzahl Patientensätze in der Datei
ROV	I	Bucketüberlaufrate (vorgegeben)
ASD	I	Anzahl belegter Spuren (ASD=APS/SBK)
AZD	I	Anzahl Zylinder der Datei (AZD=ASD/ASZ)
ZAM	I	Zylinderabstandsmittel Primär-/Überl.bereich
	I	ZAM=AZD/2
SBK	I	Spurblockkapazität 13165/(135+PSL)
STi	I	Steigung Zylinderüberquerungsgerade
AWi	I	Anfangswert Zylinderüberquerungsgerade

TABELLE 3.3

Zusätzliche Variablen für das Zugriffsmodell PATSY (PATFILE, Bild 3.10)

Modell für das Lesen der gesamten Datei (PATFILE, Seite *82*) mit GET NEXT, wobei das Herauslösen der Patientensätze aus dem Block (Bucket) im Gegensatz zu IMS vernachlässigt wird.

1. Aufsetzen auf Startzylinder (ZSZ)
2. Positionierungen in der Spur für den Primärbereich (HUPxAPS)x(1-ROV)

3. Aufsetzen, Rücksetzen Überlauf (ROVxAPSx(ZAMxSTi+AWi)x2)
4. Positionierungen in der Spur im Überlauf (HUPxAPSxROV)
5. Übertragungszeit (APSxPSL/BMS)
6. Zylinderquerungen Primärbereich (AZD-1)x(1-ROV)xZFZ)
7. Pfadzeit 1ms
8. Zeitkorrektur KORRFAK 2ms

In Zahlen:

BELG=17, APS=7648, BFAK=35, PSL=3520, LSL=100,
ALS=130000, ROV=0,05, SBK=3, ASD=2550, AZD=135, ZAM=68,
STi=ST1=0,325, AWi=AW1=10

zu 1: 40ms
zu 2: 8,35x7648x0,95=60668ms
zu 3: 0,05x7648x(68x0,325+10)x 2=24550
zu 4: 8,35x 7648x0,05=3193ms
zu 5: 7648x3520/806=33401
zu 6: (135-1)x(1-0,05)x10=1273ms
zu 7: 130000ms
zu 8: 2x130000=260000ms
Summe: 513125ms
je Patientensatz: 3,95ms

Modell für das Lesen eines einzelnen Datensatzes mit der Zugriffsfunktion GET UNIQUE:

1. Aufsetzen Startzylinder in Datei (AZDxSTi/3+AWi)
2. Positionieren in der Spur (HUP)
3. Lesezeit (PSL/BMS)
4. Überlauf: Positionieren Zylinder (ZAMxSTi+AWi)
5. Überlauf: Positionieren in der Spur (HUP)
6. Überlauf: Lesezeit (PSL/BMS)
7. Pfadzeit: 1ms
8. Zeitkorrektur KORRFAK 2ms

In Zahlen:

zu 1: 0,325x135/3+10=24,6ms
zu 2: 8,35ms
zu 3: 3520/806=4,4ms
zu 8: 2ms

Damit ist der Zugriff ohne Überlauf 39,3ms schnell.

zu 4: 68x0,325+10=32,1ms
zu 5: 8,35ms
zu 6: 3520/808=4,4ms
zu 7: 1ms

Damit ergibt sich eine Gesamtzeit für einen Zugriff mit Überlauf zu 85,2ms.

3.2.
4.1.4 IMS-HISAM

Die Organisationsform HISAM (hierarchical indexed sequential) (42) benutzt einen Primärbereich (ISAM) als erste physische Datei und einen Überlaufbereich (OSAM) als zweite physische Datei, um Datensätze zu speichern, die aus der Aneinanderreihung eines hierarchischen Baumes von Segmenten (Datensätzen) entstehen. Im MSH werden nur sogenannte "primary data set groups" verwendet. Da das Verwaltungssystem Zugriffe auf Teile (Knoten, Segmente) des Baumes zuläßt, wird auch vom gesamten Baum eines Wurzelsegmentes als logischem Datensatz gesprochen, so daß den Segmenten vom Zugriff her der Begriff Datensatz entspricht. Zum physischen Block im Primärbereich eines gespeicherten logischen Datensatzes gelangt das System über den von der ISAM-Dateiorganisation (39) her bekannten mehrstufigen, systemgepflegten Schlüsselindex (Zylinder- und Spurindex):

Wurzelsegmentschlüssel, Zylinderindex, Spurindex mit Primärbereichsadresse, diese mit Überlaufkettenadresse(n).

Der Zylinderindex enthält alle letzten Wurzelschlüssel aller Zylinder der ISAM Datei und soll sich bei der Programmausführung im Hauptspeicher befinden. Der Spurindex befindet sich in der ersten Spur eines Zylinders und wird nach dem Aufsetzen auf den betreffenden Zylinder durchsucht. Er enthält die Schlüssel des letzten Wurzelsegmentes aller Spuren eines Zylinders. Von Wedekind (98) wurde für das Lesen des Spurindexes eine Zeit von 1,5xHUP angenommen. Bei GET NEXT Zugriffen wird der Index nicht gebraucht, außer bei Spur- und Zylinderwechseln. Für die Zeitschätzung wird ein reorganisierter Zustand der Datenbank angenommen, so daß alle logischen Datensätze mit den Wurzelsegmenten im ISAM beginnen und eventuell Überlaufketten im OSAM bilden, deren Datensätze (Segmente) benachbart sind. Jedes Segment enthält die benutzerdefinierten Daten und wird um den sogenannten Präfix verlängert, der Systemdaten enthält. Dabei bekommt das Wurzelsegment grundsätzlich einen Aufschlag von 7 Byte, den abhängige Segmente nicht besitzen. Allen Segmenten wird mindestens jeweils ein Byte für die Segmentkennung und ein sogen. "Delete-Flag" hinzugefügt. (Optional sind je nach DBD-Generierung und Organisationsform noch ein 4 Byte langer Zeigerzähler und so viele Zeiger je 4 Byte, wie der Zähler angibt.)

Die Variablen für das Zugriffszeitmodell sind:

VARIAB.	I	BEDEUTUNG
PSP	I	physische Satzlänge ISAM (BLKSIZE)
PSO	I	physische Satzlänge OSAM (BLKSIZE)
LSP	I	logische Satzlänge ISAM (LRECL)
LSO	I	logische Satzlänge OSAM (LRECL)
LSL	I	logische Datensatzlänge
ALS	I	Anzahl logischer Datensätze
ZPA	I	Anzahl reservierter Zylinder ISAM
ZOA	I	Anzahl reservierter Zylinder OSAM
ZPO	I	Anzahl Zylinder zwischen ISAM/OSAM
PFXS	I	Präfixsumme zu LSL, anteilig nach Entladestatistik
KSW	I	Kanalschalter, =0:2 Kanäle, =1:1 Kanal
APS	I	Anzahl physischer Sätze ISAM (ALSxLSP/PSP)

LDS	I	Summe Datensatzlängen (ALSx(LSL+PFXS))
AOS	I	Anzahl belegter physischer Sätze im OSAM
AOD	I	Anzahl log. Datensätze, die Überläufe bilden
SBK2314	I	1+(7294-PSP)/(101+1,0435xPSP))
SBK3330	I	13165/(135+PSO)
ASI	I	belegte Spuren ISAM APS/SBK
ASO	I	belegte Spuren OSAM AOS/SBK
AZI	I	belegte Zylinder ISAM ASI/ASZ
AZO	I	belegte Zylinder OSAM ASO/ASZ
ZAM	I	Abstandsmittel ISAM/OSAM (KSWx((ZPA+AZO)/2+ZPO))
STi	I	Steigung, Zylinderquerungsgerade
AWi	I	Anfangswert, Zylinderquerungsgerade

TABELLE 3.4

Variablen des Zugriffsmodells HISAM und ihre Bedeutung.

Lesen der gesamten Datei für einen Segmenttyp mit GET NEXT:

1. Positionieren auf ISAM Anfang:ZSZ
2. Positionieren auf Zylinderanfänge
2.1 Folgezylinder im ISAM (AZI-1)xZFZ
2.2 Einstellen OSAM und Rückkehr in den ISAM
KSWx(ZAMxSTi+AWi)x2xAOD + (1-KSW)x(AZO-1)xZFZ+ZSZ)
2.3 Zylinderwechselzeit AZIx1,5x HUP
(Spurindex lesen bei Zylinderwechsel, (98))
3. Positionierungen in der Spur
3.1 ISAM APSxHUP
3.2 OSAM AODxHUP
3.3 Spurwechselzeit ASIx1,5xHUP
4. Lesezeit
4.1 ISAM LSPxALS/BMS
4.2 OSAM AODxPSO/BMS
5. Systemverwaltungszeit
5.1 Aufruf des Datenverwaltungssystems und I/O-Bereichsbearbeitung
NxZIMxIMSFAK
5.2 Pfadzeit Nx1ms

Es wird angenommen, daß jede Überlaufkette direkt eingelesen wird (AOD)
In 5.1 und 5.2 bedeutet N die Zahl der gefundenen Segmente.

Lesen von Einzelsegmenten mit der Funktion GET UNIQUE:
Hier wird eine untere Abschätzung ohne Überlaufzugriff und eine obere mit gestreutem Zugriff im Überlaufbereich getrennt angegeben. Die Datenverteilung erlaubt, Erwartungswerte für den einen oder anderen Fall anzugeben.

1. Zylinderindex (vernachlässigt, Hauptspeicher)
2. Positionieren Zylinder (AZIxSTi/3+AWi)
3. Spurindex lesen 1, 5xHUP
4. Positionierung in der Spur HUP
5. Lesen des Blockes (PSP/BMS)
6. IMS-Zeit ZIMxIMSFAK
7. Pfadzeit 1ms

8. Zylinder positionieren im Überlauf
   KSWx(ZAMxSTi+AWi) + (1-KSW)x(AZOxSTi/3+AWi)
9. Positionieren in der Spur HUP
10. Block einlesen PSO/BMS

Weitere Überlaufblöcke sind im reorganisierten Zustand unwahrscheinlich, es sei denn, die Blocklänge ist zu klein gewählt oder der logische Datensatz überlang.

Zahlenbeispiel der Datenbank NAMPAT (Bild 3.10):

```
IMSFAK=0,9+(DBL-1)/10+0,7xDBLx0,02, Platte: ISAM=2314+
OSAM=3330, PSP=3450, LSP=69, PSO=3450, LSO=69, LSL=33,
ALS=118857, ZPA=70, ZOA=20, ZPO=0, PFXS=(7+2)+2x0,13=9,26,
KSW=0, APS=2378, LDS=(33+9,26)x118857=5022897,
SBK2314=1+(7294-3450) / (1001+1,0435x3450)=2, ASI=1189,
AZI=60.
```

Problematisch sind AOS und AOD, da die Entladestatistik (unload) darüber keine Information liefert. Die naheliegende Abschätzung als AOS =(ALSx(LSL+PFXS) - ALSx LSP)/PSO ergibt im Fall des NAMPAT ein AOS =-921, was als zu viel Leerplatz im Primärbereich interpretiert werden muß und damit keine Aussage über den Überlaufbedarf ermöglicht.

Wenn die Anzahl der logischen Datensätze als Funktion der tatsächlichen Satzlängen bekannt ist, läßt sich die Überlaufkettenbildung genauer abschätzen, Bild 3.3.

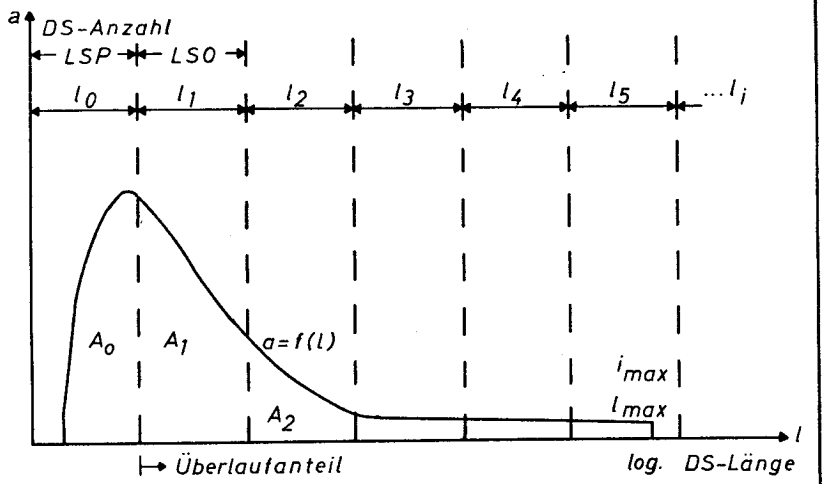


BILD 3.3

Anzahl a logischer Datensätze als Funktion der logischen Datensatzlängen.

Die Anzahl Datensätze ohne Überlaufketten ist AO, die Anzahl der Datensätze mit logischen Satzlängen im Intervall

$$\left[\sum_{o}^{n-1} l_i \quad , \quad \sum_{o}^{n} l_i \right] \quad mit \quad 1 \leq n \leq i_{max}$$

ist jeweils Ai. Der Bezug von li und Ai ist hier bewußt zu LSO gewählt, prinzipiell ist jede andere Intervallgröße möglich.

$$A_0 = \int_{o}^{l_0} a \cdot dl \quad ; \quad A_i = \int_{l_{i-1}}^{l_i} a \cdot dl \qquad mit \quad 1 \leq i \leq i_{max}$$

Ein logischer Datensatz besteht aus dem Wurzelsegment und all seinen Abhängigen. Die Länge Lmi soll die mittlere Gesamtlänge abzüglich der Primärbereichslänge LSP eines logischen Datensatzes zugehörig zu Ai sein. Damit ist das Produkt aus (Ai x Lmi) die im Sekundärbereich (Überlauf, Overflow) unterzubringende Datenlänge der zu Ai gehörenden Sätze. Falls die Überlaufsätze ohne Leerplatz aufgefüllt werden könnten, so ist die Anzahl der benötigten Überlaufsätze p in der Länge PSO:

$$p = \left(\sum_{i} A_i \cdot Lm_i \right) / PSO \qquad mit \quad 1 \leq i \leq i_{max}$$

Wenn zusätzlich bei reorganisierter Datenbank durch ausreichend großen IMS-Pufferbereich alle Überlaufsätze nur einmal eingelesen werden müssen, so ist p gleichzeitig auch die Anzahl der erforderlichen Zugriffe im Sekundärspeicherbereich. Die Größen Ai werden im MSH durch gelegentliche Auswertungen ermittelt, die den Namen "Strukturanalysen" haben. (Die Datenbank CONPAT nach Bild 3.7 hatte z.B. folgende prozentuale Verteilung der Längen der logischen Datensätze:

AO = 95 % aller logischen Sätze
A1 = 4,7 %
A2 = 0,2 %
A3 = 0,006 %.)

Wenn die Funktion a=f(l) in Bild 3.3 für jedes Intervall durch eine Gerade angenähert wird, so gilt für die mittlere Länge Lmi der logischen Datensätze dieses Intervalls:

$$lm_i = \left(\sum_{n=1}^{i-1} l_n \right) + 0{,}5 \cdot l_i$$

Damit wird die Anzahl der Zugriffe p im Oberlaufbereich abgeschätzt zu:

$$p = \left(\sum_{i=1}^{i_{max}} \left(\left(\sum_{n=1}^{i-1} l_n \right) + 0{,}5 \cdot l_i \right) \cdot A_i \right) / PSO$$

Mit li=const=LSO und BFAK=PSO/LSO ergibt sich:

$$p = (1/BFAK) \cdot \sum_{i=1}^{i_{max}} A_i \cdot (i - 0{,}5)$$

Im Fall der Datenbank NAMPAT (Bild 3.10) ergibt sich:
Zahl der Wurzelsegmente 118857 und Zahl der abhängigen NAREST Segmente 15788, also 0,13 NAREST je einem ROOT, d.h. jedes 8. ROOT besitzt ein NAREST-Segment. Bei einer ISAM-Länge von 65 Byte kann also danach kein Überlauf stattfinden. Die Verteilung hat dagegen aber in der Realität die Eigenart, daß vorwiegend entweder kein NAREST oder vier abhängig sind. Hieran wird besonders deutlich, daß die Entladestatistik für die Abschätzung des Überlaufanteiles bei fehlender Gleichverteilung oder zu geringer Segmenttypenzahl nicht geeignet ist. Jedes 30-te Wurzelsegment hat vier Segmente, so daß statt der mittleren Entladestatistik-Länge besser zwei Klassen angegeben werden. Das Bild der Funktion a=f(l) ist, Bild 3.4.

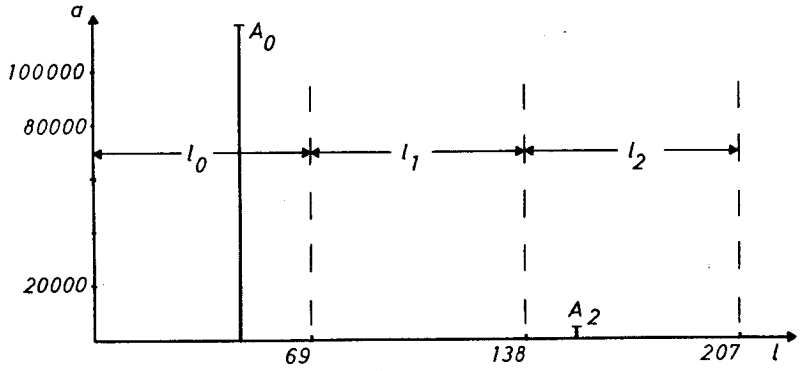


BILD 3.4

Anzahl a logischer Datensätze der Datenbank NAMPAT als Funktion ihrer Längen. l_0 = Primärbereichslänge, l_1 = Sekundärbereichslänge.

In A2 sind AOD=3947 logische Datensätze, die Überläufe in der Länge 2xLSO bilden. Die Anzahl der Überlaufblöcke ist AOS=158. Damit lassen sich die restlichen Angaben ermitteln.
SBK3330=3, ASO=53, AZO=3, ZAM=0.
Wenn nach der Formel für p gerechnet wird, so ergibt sich:

$$p = (1/50) \cdot \sum_{i=1}^{2} A_i \cdot (i - 0{,}5) = 119$$

Diese Zahl liegt gegenüber der zufällig bekannten Zahl von 158, die durch die spezielle Verteilung des einen Segmenttyps zustande kommt, um 25 % zu niedrig. Die Entladestatistik versagte ganz. Im Fall einer größeren Zahl von Segmenttypen und besserer Verteilung sollte p mit dem angegebenen Verfahren genauer sein. Eine Überprüfung war im MSH nicht möglich.

Das Lesen von nur Wurzelsegmenten ergibt eine Zeit von:

```
IMSFAK = 0,914
zu 1   :  87,5ms
zu 2.1:  25x(60-1) = 1475ms
zu 2.2:  -
zu 2.3:  60x1, 5x12,5 = 1125ms
zu 3.1:  2378x12,5 = 29725ms
zu 3.2:  -
zu 3.3:  1189x1,5x12,5 = 22294ms
zu 4.1:  118857x69/312 = 26286ms
zu 4.2:  -
zu 5.1:  7x0,914x118857 = 760447ms
zu 5.2:  1x118857ms
```

Die gesamte Datenbank wird in 960296ms gelesen. Dagegen wurde gemessen:
8,05x118857 = 956799ms, was einem Fehler von 0,4 % entspricht.

Lesen der NAREST-Segmente:

```
IMSFAK = 1,028
zu 1   :  87,5ms
zu 2.1:  (60-1)x25 = 1475ms
zu 2.2:  (3-1)x10+40 = 60ms
zu 2.3:  60x1,5x12,5 = 1125ms
zu 3.1:  2378x12,5 = 29725ms
zu 3.2:  3947x 12,5 = 49338ms
zu 3.3:  1189x1,5x12,5 = 22294ms
zu 4.1:  69x118857/312 = 26286ms
zu 4.2:  3947x3450/806 = 16895ms
zu 5.1:  7x1,028x15788 = 113610ms
zu 5.2:  15788ms
Summe :  276683ms
gemessen:   17,08x15788 = 269659ms, Fehler: 2,6 %.
```

Gestreuter Zugriff auf ein Wurzelsegment mit der Funktion GET UNIQUE:

1. kein Überlaufanteil
2. 60x0,45/3+45 = 54ms
3. 12,5x1,5 = 18,75ms
4. 12,5ms
5. 3450/312 = 11ms
6. 7x0,914 = 6,4 ms
7. 1ms

Summe für ein Segment: 103,5ms

Zahlenbeispiel Datenbank AUFPAT (Bild 3.10):

IMSFAK = 0,907
LSO=LSP=21, PSO=PSP=3465, LSL=18, PFX=3, ALS=139144,
ZPA=30, ZOA=1, ZPO=0, KSW=0, LDS=2922024,
APS=139144x21/3465=844, AOS=0, AOD=0, ASI=844/SBK=422,
SBK=1+(7294-3465)/(101+1,0435x3465)=2, AZI=422/20=22, AZO=0,
ZAM=0.
Wurzelsegmente, GET NEXT, kein Überlauf.

1. : 87,5ms
2.1: (22-1)x25 = 525ms
2.2: -
2.3: 22x1,5x12,5 = 412,5ms
3.1: 844x12,5 = 10550ms
3.2: -
3.3: 422x1,5x12,5 = 7913ms
4.1: 21x139144/312 = 9366ms
4.2: -
5.1: 139144x7x0,907 = 883425ms
5.2: 139144ms

Summe: 1051422, Einzelsegment: 7,6ms

Wurzelsegment, GET UNIQUE:

1. : -
2. : 22x0,45/3+45 = 48,3ms
3. : 1,5x12,5 = 18,75ms
4. : 12,5ms
5. : 3465/312 = 11,1ms
6. : 6,4ms
7. : 1ms

Summe: 98ms

3.2.4.1.5 IMS-HDAM

Die Organisationsform HDAM benutzt eine Adreßprozedur, die zum Schlüssel eines Wurzelsegmentes eine relative Block- und Ankerpunktnummer errechnet und so die physische Speicherung festlegt. Diese Prozedur muß vom Datenbankverwalter (DBA) gestellt werden. Die Daten befinden sich in einer Datei, die wie der OSAM bei der Zugriffsform HISAM direkt adressiert ist. Die OSAM-Datei ist allerdings in zwei

Bereiche unterteilt: der adressierbare Bereich (root addressable area) und der verkettete Bereich (overflow area, Überlauf). Die errechnete Blocknummer weist auf eine Bolckadresse des adressierbaren Bereiches. Dort können Segmente des logischen Datensatzes beim Laden in einer per Datenbankdefinition (DBD) maximal vorgegebenen Länge abgelegt werden. Darüber hinausgehende Daten werden im verketteten Bereich der Datei untergebracht. Bei normalen Änderungsvorgängen wird versucht, Segmente in der Nähe der Blocknummer des adressierten Bereichs in freien Plätzen unterzubringen. Die Zahl der Zylinder, die davor oder danach für Freiplatzsuche (SCAN) erlaubt sind, wird in der DBD festgelegt. Dadurch wird zwar der Aufwand beim Hinzufügen größer, der Suchaufwand wegen geringerer Überläufe jedoch günstiger. Diese Maßnahmen greifen allerdings nur dann, wenn der adressierbare Bereich groß genug ist.

Benötigte Variablen für das Zugriffszeitmodell HDAM sind:

VARIAB.	I	BEDEUTUNG
PSO, PSP	I	Physische Satzlänge im OSAM (BLKSIZE)
ALS	I	Anzahl logischer Datensätze (Entladestatistik)
ZPA	I	Anzahl Zylinder adressierbarer Bereich
ZOA	I	Anzahl Zylinder verketteter Bereich
LSL	I	logische Datensatzlänge (Entladestatistik)
LSP	I	logische Satzlänge in adress. Bereich, rechnerisch
LSO	I	logische Satzlänge in verkettetem Bereich
PFXS	I	Präfixsumme (analog HISAM)
APS	I	Anzahl phys. Sätze, adressierbarer Bereich
AOS	I	Anzahl phys. Sätze in verkettetem Bereich
AOD	I	Anzahl log. Datensätze mit Überlaufkette
ADC	I	Anzahl Datensätze mit SCAN nach DBD
APZ	I	Ankerpunktzahl in Rechnungen
LDS	I	Länge der belegten Datei (ALSx(LSL+PFXS))
SBK3330	I	13165/(135+PSP)
ASI	I	belegte Spuren adressierbarer Bereich (APS/SBK)
ASO	I	belegte Spuren verketteter Bereich (AOS/SBK)
AZI	I	belegte Zylinder adressierbarer Bereich (ASI/ASZ)
AZO	I	belegte Zylinder verketteter Bereich (ASO/ASZ)
ZAM	I	Zylinderabstandsmittel ((ZPA+AZO)/2)
APLNR	I	APNRxAPBP/APTP
APNR	I	laufende Ankerpunktnummer
APBP	I	Mittlerer Platzbedarf, Daten zu einem AP
APTP	I	Tatsächlicher Platz bei einem Ankerpunkt (AP)
SCAN	I	adressierbarer Bereich, max. Abstand Daten von AP
	I	gemäß Datenbankbeschreibung (DBD)

TABELLE 3.5

Variablen für das Zugriffsmodell HDAM und ihre Bedeutung.

Die letzten fünf Variablen werden bei der Durchrechnung des Beispieles für die Datenbank APAT ausführlich behandelt. Die Ähnlichkeit von Bezeichnungen zu HISAM bei Variablen ist bewußt gewählt, da sich ähnliche Zeitbetrachtungen ergeben.

Lesen aller logischen Datensätze mit allen Segmenten, GET NEXT und ohne Adressierungsprozedur, da TWIN-Verkettung auf Wurzelsegmentebene vorliegt:

1. Positionieren auf Dateianfang (ZSZ)
2. Positionieren auf Zylinderanfänge
2.1 Folgezylinder im adressierbaren Bereich (APLNR+1)xZFZ
2.2 SCAN-Bereich: (nxSTi+AWi)xADC
2.3 Ü. Zylinder und Rückkehr in adressierbaren Bereich (ZAMxSTi+AWi)x2xAOS
3. Positionierungen in der Spur
3.1 Adressierbarer Bereich (APSxHUP)
3.2 SCAN-Bereich: HUPxADC
3.3 Verketteter Bereich (AOSxHUP)
4. Lesezeit
4.1 Adressierbarer Bereich ca. (APSxPSP/BMS)
4.2 SCAN-Bereich: (PSO/BMS)xADC
4.3 Verketteter Bereich: (AOSxPSO/BMS)
5.1 IMS Datenverwaltungssystemzeit NxIMSFAKxZIM
5.2 Pfadzeit Nx1 ms

In 2.2, 3.2 und 4.2 soll der Anteil von ADC Datensätzen für die Suche +n und -n Zylinder vom Ankerpunkt entfernt gemäß DBD-Angabe (SCAN) berücksichtigt werden. Dabei ist ADC zu schätzen.
Der Faktor AOS in 2.3, 3,3 und 4.3 drückt aus, daß vollständige verkettete log. Datensätze im verketteten Bereich untergebracht werden und nicht Restlängen.

Lesen einzelner Segmente mit GET UNIQUE:
Der Lesekopf steht im Bereich der Datenbank. Die Zeitanteile werden in drei Gruppen geteilt: nur ein Block im adressierbaren Bereich, Überlauf in benachbarte Zylinder und Überlauf in den verketteten Bereich.

1. Adresse ermitteln (vernachlässigt)
2. Zylinderpositionieren (AZI+AZO)xSTi/3+AWi
3. Positionierung in der Spur HUP
4. Lesen Block (PSP/BMS)
5.1 IMS-Zeit IMSFAKxZIM
5.2 Pfadzeit 1ms

Überlauf in Nachbarzylinder (SCAN n)

6a. Zylinderpositionierung (nxSTi+AWi)
7a. Spurpositionierung (HUP)
8a. Lesen Block (PSO/BMS)

Überlauf in den verketteten Bereich

6b. Zylinder positionieren (ZAMxSTi+AWi)
7b. Positionieren in Spur (HUP)
8b. Lesen Block (PSO/BMS)

Zahlenbeispiel, Datenbank APAT (Bild 3.9):

IMSFAK=0,9*(DBL-1)/10+0,7xDBLx0,14; Platte=3330,
PSP=PSO=3008, ALS=65315, LSL=781, PFXS=220, LDS=65400000,
SBK=13165/((135+3008)=4, APS=12000, LSP=LSL, LSO=LSL,
ASI=3000, AZI=158, ZPA=158, ZOA=142, AOS=9772, AOD=29315,
ASO=2443, AZO=129, ZAM=144.

Die Adreßberechnung für die Datenbank APAT erzeugt aus Wurzelsegmentschlüsseln bis zu 12000 Ankerpunktadressen. Die den Ankerpunkten unmittelbar zugeordneten physischen Sätze (Blocknummern) bilden den adressierbaren Bereich. Die Summe der mittleren Satzlängen ergibt für die einem Ankerpunkt zugeordneten logischen Datensätze im allgemeinen eine größere Länge, als der verfügbare Platz groß ist.

Ein Extrem für die Speicherauslegung ist, die Blocklänge ausreichend groß zu wählen und die Anzahl der Byte pro logischem Datensatz beim Laden so zu beschränken, daß Restlängen in Freiplätzen der näheren (SCAN-) Umgebung des Ankerpunktes untergebracht werden können und selten Restlängen in den verketteten Bereich auszulagern sind.

Das andere Extrem ist, die Anzahl der Byte pro logischem Datensatz beim Laden nicht zu beschränken, so daß unterbringbare logische Datensätze vollständig in Freiplätzen der über DBD definierten SCAN-Zylinder untergebracht werden und nicht mehr unterbringbare logische Datensätze vollständig im verketteten Bereich gespeichert werden.

Das erste Extrem bringt gute Suchzeiten und schlechtere Speicherausnutzung. Das zweite Extrem bringt höchste Speicherausnutzung, aber schlechtere Suchzeiten. Im MSH wurde das zweite Extrem dadurch angenähert, daß beim Übergang von HISAM auf H_DAM-Organisation die Satzlängen der logischen Datensätze für den adressierbaren Bereich nicht per DBD beschränkt wurden. Für die Datenbanken APAT und LEIPAT ergibt sich hieraus die Abschätzung der Anteile im verketteten Bereich und der SCAN-Umgebung.

In der Datenbank APAT gibt es sechs bis sieben logische Datensätze, die auf denselben Ankerpunkt weisen. Platz ist im physischen Block eines Ankerpunktes für drei mittlere Satzlängen. Grob entspricht einem Ankerpunkt der Datenplatz von zwei Ankerpunktblöcken, d.h. ein Zylinder mit seinen Ankerpunkten braucht für die Speicherung der zugehörigen Daten noch einen weiteren Zylinder zusätzlich. Durch die Begrenzung der Entfernung vom AP-Zylinder für die Datenunterbringung, muß ab einem bestimmten Zylinder ein Teil der Daten im verketteten Bereich abgelegt werden. Dies führt im MSH für die reorganisierte Datenbank etwa zu einer Verteilung nach Bild 3.5. Zur Verfügung steht bei den Ankerpunkten nur etwa die Hälfte des benötigten Platzes.

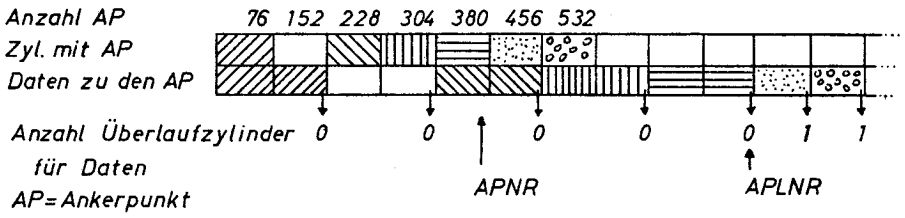


BILD 3.5

Speicherbelegung und Überlaufanteile für die HDAM-Organisation der Datenbank APAT. Nähere Erläuterungen folgen im Text.

Da die unmittelbar aufeinanderfolgenden Zylinder in 2.1 und die Anzahl der Lesebewegungen in den Überlaufbereich in 2.3 in die Zeitrechnungen eingehen, soll eine Abschätzung angegeben werden, die es erlaubt, die benötigten Größen zu ermitteln. (Das Verfahren gilt nur, wenn die logischen Satzlängen im adressierbaren Bereich wie im MSH nicht begrenzt sind.)

Ermittelt werden soll der Zylinder APLNR und APNR nach Bild 3.5 sowie die Anzahl der erforderlichen Kammbewegungen in den verketteten Bereich bei sequentiellem Lesen. Dabei sind folgende Variablen beteiligt:

APTP = Tatsächlich vorhandener Platz für die zu einem Ankerpunkt (AP) gehörenden Daten in Byte,

APBP = Im Mittel benötigter Platz für die zu einem AP gehörenden Daten in Byte,

SCAN = Zugelassene Entfernung vom Ankerpunkt in Zylindern für die Speicherung der zu diesem AP gehörenden Daten im adressierbaren Bereich,

APLNR = Letzter vollständig belegbarer Zylinder im adressierbaren Bereich mit Daten, die zu den Ankerpunkten des Zylinders APNR gehören (Bild 3.5)

APLNR=APNRxAPBP/APTP gilt unter der Bedingung, daß APNR+SCAN kleiner als APLNR ist.
Durch Umformung erhält man die Bedingung für APLNR:

$$APLNR \leq \frac{SCAN}{1 - APTP/APBP} \quad , \quad APTP < APBP \quad , \quad SCAN > 0$$

Der tatsächliche Platz APTP entspricht der Angabe PSP, der benötigte Platz ist ungefähr
APBP=(Anzahl Wurzelsegmente/Max. AP-Zahl) x (LSL+PFXS).
Mit APNR=(APLNR-SCAN) ist der Zylinder mit Ankerpunkten ermittelbar, dessen zugehörige Daten noch vollständig in der SCAN-Entfernung untergebracht werden können. Die Datenmenge, die von APLNR an noch in der SCAN-Entfernung von den AP-Adressen speicherbar ist, ist jeweils ein Zylinder. Die im verketteten Bereich abzulegende Menge an Daten ist APBP/APTP-1 Zylinder.

Beim sequentiellen Lesen über Twin-Verkettung können APLNR+1 Zylinder ohne Überlaufzugriffe gelesen werden, dann erfolgt jeweils im Wechsel der Sprung in den verketteten Bereich für APBP/APTP-1 Zylinder, dann Rücksprung für 1 Zylinder, u.s.w. Die Anzahl der erforderlichen Hin- und Rücksprünge entspricht ungefähr der Anzahl der Zylinder im adressierbaren Bereich, die ab APLNR+1 noch zu lesen sind:
(Gesamt zu lesende Zyl. -(APLNR+1)xAPTP/APBP

Das Bild 3.5 geht von einem Verhältnis APTP/APBP = 0,5 aus, in Wirklichkeit ist APTP/APBP = 0,55. Deshalb ist die Zeichnung, die der Veranschaulichung dient, ungenauer als die Rechnung.

Datenbank APAT:
APBP = (65315/12000) x (781 + 220) = 5449
APTP = 3008, APLNR = 11, APNR = 6.

Lesen der Wurzelsegmente APROOT:
IMSFAK = 0,998.
8000 Segmente mit GET NEXT, d.h. ohne über die Ankerpunkte zu lesen, sequentiell durch die Blöcke der Datei. SCAN-Zeiten treten nicht auf, wohl aber Oberlaufzeiten. Da sich logische Datensätze komplett im verketteten Bereich befinden, erfolgt ein Hin- und Rücksprung nicht bei jedem der AOS Sätze, sondern nach dem angegebenen Verfahren in etwa 13 Fällen.

```
zu 1.   :  40ms
zu 2.1 :  (11+1)x10=120ms
zu 2.2 :  -
zu 2.3 :  (144x0,325+10)x2x13=1477ms
zu 3.1 :  25 Zyl = 1900 Sätze,   1900x8,35=15865ms
zu 3.2 :  -
zu 3.3 :  10 Zyl =  760 Sätze,    760x8,35= 6346ms
zu 4.1 :  1900x3008/806 = 7091ms
zu 4.2 :  -
zu 4.3 :  760x3008/806 = 2836ms
Summe 1-4:     33775ms
zu 5.1 :  7x0,998x8000=55888ms
zu 5.2 : 8000ms
Summe, total: 97663ms,
gemessen 92720ms,  Fehler 5,3 %.
```

Lesen Segmenttyp APLDAT:
Gegenüber der Unloadstatistik (Bild 3.8) kann angenommen werden, daß etwa 30 % weniger Labordaten bei den älteren Geburtsjahrgängen vorliegen, als die mittlere Verteilung der Entladestatistik ausweist. Beim sequentiellen Lesen werden die ältesten Patienten wegen der Sortierung nach dem Geburtsdatum zuerst gelesen. So sollen ca. 0,7 APLDAT Segmente je Patient im Mittel auftreten. Damit sind die Daten von 11429 Patienten zu prüfen, was etwa 50 Zylindern entspricht. Anzahl Sprünge in den verketteten Bereich: 21.

```
IMSFAK = 1,196
zu 1.   :  40ms
zu 2.1 :  12x10=120ms
zu 2.2 :  -
zu 2.3 :  (144x0,325+10)x2x21=2386ms
zu 3.1 :  33 Zyl = 2508 Sätze,   2508x8,35=20942ms
zu 3.2 :  -
zu 3.3 :  17 Zyl = 1292 Sätze,   1292x8,35=10788ms
zu 4.1 :  2508x3008/806= 9360ms
zu 4.2 :  -
zu 4.3 :  1292x3008/806=4822ms
Summe 1-4:     48458ms
zu 5.1 :  7x8000x1,196=66976ms
zu 5.2 :  8000ms
Summe, total:  123433ms
gemessen: 15,95x8000=127600ms,   Fehler 3,3 %.
```

Lesen APWERT Segmente:
IMSFAK = 1,394

Zu lesen sind zu 1 bis 4 nur ca. 1/12 der Datenmenge der APLDAT Segmente (Entladestatistik, Bild 3.8)

```
1.-4.  :  48458/12 = 4038ms
zu 5.1 :  7x8000x1,394=78064ms
zu 5.2 :  8000ms
Summe  :  90102
gemessen: 11,07x8000=88560ms,   Fehler 1,7 %.
```

Lesen APBE Segmente:
IMSFAK = 1,196

Datenmenge wie für Wurzelsegmente

```
1.-4.  :  33775ms
zu 5.1 :  7x8000x1,196=66976ms
zu 5.2 :  8000ms
Summe  :  108751ms
gemessen: 14,26x8000=114080ms,   Fehler 4,7 %.
```

Lesen von APARZT Segmenten:
IMSFAK = 1,394

Die älteren Jahrgänge haben weniger Behandlungen und damit weniger APARZT Segmente. Es werden zwei APARZT Segmente pro logischem Datensatz angenommen, d.h. zu 1 bis 4 ist die Hälfte der Wurzelsegmentzeit anzunehmen.

```
1.-4.  :  33775/2=16888ms
zu 5.1 :  7x8000x1,394=78064ms
zu 5.2 :  8000ms
Summe  :  102952ms
gemessen: 8000x12,28=98240ms,   Fehler 4,6 %.
```

Lesen der Wurzelsegmente APAT im Direktzugriff:

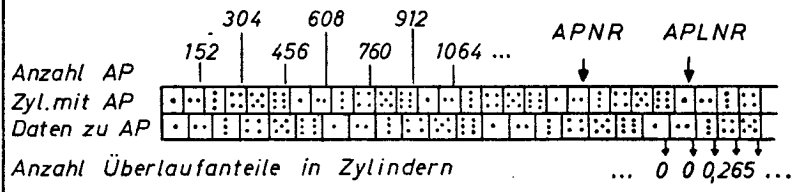
```
zu 1.  :  -
zu 2.  :  (158+129)x0,094/3+17,5)=26,5ms
zu 3.  :  8,35ms
zu 4.  :  3008/806=3,7ms
zu 5.1 :  7ms
zu 5.2 :  1ms
Ohne SCAN-Bereich und ohne verketteten Bereich: 46,6ms
zu 6a  :  5x0,325+10=11,6ms
zu 7a  :  8,35ms
zu 8a  :  3,7ms
Mit SCAN-Bereich + 5 Zylinder: 70,2ms
zu 6b  :  144x0,325+10=56,8ms
zu 7b  :  8,35ms
zu 8b  :  3,7ms
Mit verkettetem Bereich:  115,4ms
```

Zahlenbeispiel, Datenbank LEIPAT

Für die Durchrechnung der Datenbank LEIPAT (Bild 3.8) gelten zu APAT analoge Betrachtungen:

APBP = (62980/12000) × (642 + 83) = 3305
APTP = 3008, APLNR = 23, APNR = 18
IMSFAK = 0,9 +(DBL - 1)/10 + 0,7 × 7 × DBL/100,
Platte = 3330, PSP = PSO = 3008, ALS = 62980, ZPA = 158, LSL = 642,
PFXS = 83, LDS = 45660500, APS = 12000, LSP = LSL, LSO = LSL,
ASI = 3000, AZI = 158, ZOA = 42, ASO = 798, AOS = 3180, SBK = 4,
AOD = (42/158) × 62980 = 16742 (logische Datensätze liegen vollständig im verketteten Bereich), ASO = 795, AZO = 42, ZAM = 100.

Pro physischem Satz sind vier logische Datensätze speicherbar, pro Ankerpunkt müssen aber 5,25 logische Datensätze gespeichert werden. Daraus ergibt sich Bild 3,6 für die Speicherbelegung:



AP = Ankerpunkt
'SCAN' Zyl. Anzahl: 5

BILD 3.6

Speicherbelegung und Überlaufanteile für die HDAM-Organisation der Datenbank LEIPAT. Die Bedeutung ist analog Bild 3.5 gegeben und dort im Text erklärt. (S.66/67)

Lesen der Segmente LEROOT mit GET NEXT (25,4 Zylinder an Daten):

IMSFAK = 0,949
zu 1. : 40ms
zu 2.1 : 240ms
zu 2.2 : -
zu 2.3 : (100 x 0,325 + 10) x 2 x 1,1 = 94ms
zu 3.1 : 25,1 Zyl = 1908 Sätze, 1908 x 8,35 = 15932ms
zu 3.2 : -
zu 3.3 : 0,3 Zyl = 23 Sätze, 23 x 8,35 = 192ms
zu 4.1 : 1908 x 3008/806 = 7121ms
zu 4.2 : -
zu 4.3 : 23 x 3008/806 = 86ms
Summe 1-4: 23705ms

zu 5.1 : 7 x 8000 x 0,949 = 53144ms
zu 5.2 : 8000ms
Summe, total: 84849ms,
gemessen 10,45 x 8000 = 83600ms, Fehler 1,5 %.

Lesen Segmente LEAUFN mit GET NEXT:
Ca. 1,38 LEAUFN je Wurzelsegment, d.h. 0,72 des Leseaufwandes wie für Wurzelsegmente (Bild 3.8).
IMSFAK = 1,098

Summe 1-4 : wie 0,72 von LEROOT: 17068ms
zu 5.1 : 7 x 1,098 x 8000 = 61488ms
zu 5.2 : 8000ms
Summe, total: 86556ms
gemessen 11,1 x 8000 = 88800ms, Fehler 2,5 %.

Lesen LEKLI Segmenttyp, GET NEXT:
Ca. 1,93 LEKLI je LEROOT, d.h. 0,52 Leseaufwand wie für Wurzelsegmente (Bild 3.8).
IMSFAK = 1,247

zu 1. : 40ms
Summe 1-4 : wie 0,52 von LEROOT: 12327ms
zu 5.1 : 7 x 8000 x 1,247 = 69832ms
zu 5.2 : 8000ms
Summe 90159ms
gemessen 11,7 x 8000 = 93600ms, Fehler 3,7 %.

Lesen LEROOT im Direktzugriff, GET UNIQUE:
IMSFAK = 0,949

zu 1. : -
zu 2. : 158 x 0,325/3 + 10 = 27ms
zu 3. : 8,35ms
zu 4. : 3008/806 = 3,73ms
zu 5.1 : 7 x 0,949 = 6,6ms
zu 5.2 : 1ms
Zugriffszeit ohne SCAN- und verketteten Bereich: 46,8ms
zu 6a : 5 x 0,325 + 10 = 11,63ms
zu 7a : 8,35ms
zu 8a : 3,73ms
Zugriffszeit mit SCAN-Bereich + 5 Zylinder: 70,5ms
zu 6b : 100 x 0,325 + 10 = 42,5ms
zu 7b : 8,35ms
zu 8b : 3,73ms
Zugriffszeit mit verkettetem Bereich: 101,4ms.

3.2.4.1.6 IMS - HIDAM

HIDAM läßt sich am besten im Vergleich zu HDAM erklären:

- *Die Adreßprozedur von HDAM entfällt. Für einen Direktzugriff wird die physische Blocknummer nicht über einen Algorithmus errechnet, sondern einer systemverwalteten Schlüsseldatenbank entnommen. Diese enthält die Wurzelsegmentschlüssel und physischen Blocknummern. Das Antwortzeitverhalten bei GET UNIQUE wird wegen des HISAM Anteiles gegenüber HDAM langsamer.*
- *Die Datensätze befinden sich in einer HDAM Datei, mit dem Unterschied, daß keine Einteilung in adressierbaren Bereich und verketteten Bereich stattfindet. Dadurch wird die Antwortzeit gegenüber HDAM bei GET NEXT und reorganisiertem Datenbestand günstiger.*

Benötigte Variablen sind für die Indexdatenbank wie im Modell HISAM erforderlich und für die Datenspeicherung ähnlich dem Modell HDAM. Es wird angenommen, daß die Indexdatenbank von der Daten-Datenbank durch zwei Kanäle getrennt ansprechbar ist. Der OSAM der Indexdatenbank wird nicht betrachtet (reorganisiert).

Index-Datenbank

VARIABLE	I	BEDEUTUNG
LSP	I	log. Satzlänge (LRECL=LSP=PFXS1+KEYL)
PSP	I	physische Satzlänge (BLKSIZE)
BFAK	I	Blockungsfaktor PSP/LSP
ZPA	I	Anzahl Zylinder ISAM
PFXS1	I	Präfixsumme (9, weil nur Wurzelsegment)
APS	I	Anzahl physischer Sätze
SBK2314	I	1 + (7294 - PSP)/(101 + 1,0435 x PSP)
ASI	I	belegte Spuren ISAM APS/SBK2314
AZI	I	belegte Zylinder ISAM ASI/ASZ
KEYL	I	Schlüsselfeldlänge Wurzel Daten-DB

Daten-Datenbank

VARIABLE	I	BEDEUTUNG
PSO	I	physische Satzlänge (BLKSIZE = LRECL)
ALS	I	Anzahl log. Datensätze (Entladestatistik)
ZOA	I	Anzahl reservierter Zylinder
LSL	I	log. Datensatzlänge (Entladestatistik)
PFXS2	I	Präfixsumme
AOS	I	Anzahl physischer Sätze (ALSx(LSL+PFXS2))/PSO
LDS	I	Länge der Daten ALS x (LSL + PFXS2)
SBK3330	I	13165/(135 + PSP)
ASO	I	Anzahl belegter Spuren AOS/SBK3330
AZO	I	Anzahl belegter Zylinder ASO/ASZ
AOD	I	Anzahl Datensätze, deren log. Datensätze
	I	aus Teilen mit Kettenverweisen bestehen

TABELLE 3.6

Variablen für das Zugriffsmodell HIDAM und ihre Bedeutung.

Zugriffsmodell für GET NEXT:

1.	Indexdatenbank (nur ISAM)
1.1	Positionierung auf Dateianfang
1.2	Positionierung auf Zylinderanfänge
1.2.1	Zylinderfolgezeiten (AZI - 1) x ZFZ
1.2.2	OSAM (nicht betrachtet)
1.2.3	Zylinderwechselzeit AZI x 1,5 x HUP
1.3	Positionierungen in der Spur
1.3.1	APS x HUP
1.3.2	(OSAM)
1.3.3	Spurwechselzeit ASI x 1,5 x HUP
1.4	Lesezeit
1.4.1	PSP x APS/BMS
1.4.2	(OSAM)
2.	Daten-Datenbank
2.1	Positionierung auf Dateianfang ZSZ
2.2	Positionierung auf Zylinderanfänge
2.2.1	Folgezylinder (AZO - 1) x ZFZ)
2.2.2	Überlaufketten (AZO x 0,5 x STi + AWi) x 2 x AOD
2.3	Positionierungen in der Spur
2.3.1	Spuren ASO x HUP
2.3.2	Überlaufketten AOD x HUP
2.4	Lesezeiten
2.4.1	AOS x PSO/BMS
2.5.1	IMS Verwaltungszeit ZIM x IMSFAK x N
2.5.2	Pfadzeit 1 x N

Lesen von Segmenten mit GET UNIQUE:

1.	Index-Datenbank
1.1	Zylinderindex vernachlässigt
1.2	Positionieren Zylinder (AZI x STi/3 + AWi)
1.3	Spurindex 1,5 x HUP
1.4	Spurpositionierung HUP
1.5	Lesezeit PSP/BMS
2.	Daten-Datenbank
2.1	Positionieren Zylinder (AZO x STi/3 + AWi)
2.2	Spurpositionierung HUP
2.3	Lesezeit PSO/BMS
2.4	IMS Verwaltungszeit ZIM x IMSFAK
2.5	Pfadzeit 1ms

Zahlenbeispiel, Datenbank CONPAT (Bild 3.7):

IMSFAK = 0,9 + (DBL -1)/10 + 0,7 × 0,13 DBL, PSO = 3004, ALS = 120575,
ZOA = 280, LSL = 313, PFXS2= 76, AOS = 15614, LDS = 46903675,
SBK3330 = 4, ASO = 3904, AZO = 206, AOD = 0, LSP = 13, PSP = 3471,
BFAK = 267, ZPA = 20, PFXS1= 9, APS = 452, SBK2314 = 2, KEYL = 4,
ASI = 226, AZI = 12.

Lesen der COROOT Segmente mit GET NEXT (alle Segmente):
IMSFAK = 0,991

zu 1.1 : 87,5ms
zu 1.2.1 : (12 - 1) x 25 = 275ms
zu 1.2.3 : 12 x 1,5 x 12,5 = 225ms
zu 1.3.1 : 452 x 12,5 = 5650ms
zu 1.3.3 : 226 x 1,5 x 12,5 = 4237,5ms
zu 1.4.1 : 3471 x 452/312 = 5028,5ms
zu 2.1 : 40ms
zu 2.2.1 : (206 - 1) x 10 = 2050ms
zu 2.3.1 : 3904 x 8,35 = 32598,4ms
zu 2.4.1 : 15614 x 3004/806 = 58194ms
zu 2.5.1 : 7 x 120575 x 0,991 = 836429ms
zu 2.5.2 : 120575ms
Summe 1065391ms,
gemessen 120575 x 9,06 = 1092410ms, Fehler 2,5 %.

Lesen der COBEHA Segmente mit GET NEXT (8000 Segmente):
IMSFAK = 1,182

zu 1. : ca. 1,2 COBEHA je Wurzelsegment, d.h. 0,055 des Wurzelsegmentanteiles = 852,7ms
zu 2.1-2.4: 92882 x 0,055 = 5109ms
zu 2.5.1 : 7 x 8000 x 1,182 = 66192ms
zu 2.5.2 : 8000ms
Summe 80154ms,
gemessen 11,01 x 8000 = 88080ms, Fehler 9 %.

Lesen der COORT Segmente mit GET NEXT:
IMSFAK = 1,182

Zwei COORT je Wurzelsegment, d.h. Anteile 4000/120575
zu 1. : (4000/120575) x 15504 = 514ms
zu 2.1-2.4: (4000/120575) x 92882 = 3081ms
zu 2.5.1 : 7 x 8000 x 1,182 = 66192ms
zu 2.5.2 : 8000ms
Summe 77787ms,
gemessen 10,3 x 8000 = 82400, Fehler 5,6 %.

Lesen der COQUAL Segmente mit GET NEXT:
IMSFAK = 1,373
Statt 58 % sind für die älteren Geburtsjahrgänge nur etwa 30 % COQUAL Segmente bezogen auf Wurzelsegmente vorhanden. Daraus folgt, daß etwa Daten von 26667 Patienten zu durchsuchen sind. Der hohe Fehler von 15 % kann nur durch die Ungenauigkeit dieser Schätzung erklärt werden.

zu 1. : (26667/120575) x 15504 = 3429ms
zu 2.1-2.4: (26667/120575) x 92883 = 20542ms
zu 2.5.1 : 7 x 8000 x 1,373 = 76888ms
zu 2.5.2 : 8000ms
Summe 108859ms,
gemessen 16,07 x 8000 = 128560ms, Fehler 15 %.

Lesen der CODIKO Segmente mit GET NEXT:
Bei den älteren Geburtsjahrgängen soll wie bei COQUAL eine Minderung der Häufigkeit um 28 % angenommen werden, so daß statt 208 % noch 180 % CODIKO auf die Wurzelsegmente entfallen. Damit sind 4444 logische Datensätze zu durchsuchen.
IMSFAK = 1,564

zu 1. : (4444/120575) x 15504 = 571ms
zu 2.1-2.4: (4444/120575) x 92882 = 3423ms
zu 2.5.1 : 7 x 8000 x 1,564 = 87584ms
zu 2.5.2 : 8000ms
Summe 99578
gemessen 14,8 x 8000 = 118400ms, Fehler 16 %.

Lesen eines COROOT Segmentes mit GET UNIQUE:

zu 1.1 : -
zu 1.2 : 12 x 1,6/3 + 25 = 31,4ms
zu 1.3 : 1,5 x 12,5 = 18,75ms
zu 1.4 : 12,5ms
zu 1.5 : 3471/312 = 11,1ms
zu 2.1 : 206 x 0,094/3 + 17,5 = 23,95ms
zu 2.2 : 8,35ms
zu 2.3 : 3904/806 = 4,84ms
zu 2.4 : 7 x 0,991 = 6,9ms
zu 2.5 : 1ms
Summe 118,8ms

3.2.4.1.7 SURE

Der Suchrechner kennt die Phase der Trefferlistenerstellung mit der Suchelektronik und die Phase des Einlesens der Treffersätze im Direktzugriff. Die Datensätze der Relationen befinden sich im allgemeinen ungeordnet auf dem Speicherplatz verteilt, so daß bestimmte Sortierfolgen in einem weiteren Schritt im Suchrechner oder Hauptrechner erzeugt werden müssen. Bei wenigen Treffern kann dies im Hauptspeicher geschehen, bei größeren Treffermengen kann versucht werden, die SURE Möglichkeiten hierfür auszunutzen (53). Wünschenswert ist die Bereitstellung eines Sortierprozessors oder anderer Funktionen (53), auch für die schrittweise Erarbeitung von Suchergebnissen. Bei Zugriffen in Relationen werden ganz allgemein Bedingungen an Attributwerte, die z.B. in einer Normalform der Aussagenlogik definiert wurden, in Mengenoperationen von Treffermengen formal aufgelöst:

Der Konjunktion zweier Bedingungen an Attributwerte entspricht die Schnittmenge der beiden Treffermengen und der Disjunktion zweier Bedingungen entspricht die Vereinigungsmenge der beiden Treffermengen. Bei der richtigen Wahl der Reihenfolge der Treffermengen und Bildung von Zwischenergebnissen können Zeitoptimierungen erreicht werden. Auch

kann z.B. bei der Konjunktion gegebenfalls die Mengenoperation durch eine gezielte Suche mit Attributbedingungen die Erstellung der Treffermenge einsparen. Diese Optimierungen sind anfrage-, dateninhalte- und datenverteilungsabhängig und setzen eine entsprechend intelligente Sprachübersetzung voraus. Zu diesem Optimierungsprozeß werden keine allgemeinen Aussagen gemacht, jedoch wird bei den einzelnen Arbeitslasten versucht, die günstigste Verarbeitung zu verwenden.

Die Blockung der Datensätze einer Relation entscheidet mit über die Lesegeschwindigkeit. So gilt bei sequentiellem Lesen bei hoher Blockung eine gegenüber kleiner Blockung günstigere Lesezeit. Da beim Suchrechner aber grundsätzlich keine Schlüsselorientierung vorauszusetzen ist (53), spielt die Blockung nicht diese Rolle, sie erhöht im Gegenteil unnötig die zu bewegenden Datenmengen beim Lesen. Deshalb wird für die Datensätze nicht Spurblockung, sondern eine Blockgröße von $n \times 256$ mit $2 \leq n \leq 5$ vorgesehen (Blocklänge in den Suchzylindern (91)).

Das Benutzerhandbuch für die Grundsoftware (91) sieht folgende Unterprogramme für Datenbankanfragen vor:

- *Benutzeranmeldung SRAN*
- *Datenbankeröffnung OEFF*
- *Eingabe der Suchrechnersyntax an den Vorübersetzer UEB für die interne Darstellung*
- *Übergabe eines Suchauftrages an die Suchelektronik SUAUF*
- *Einholen von Trefferergebnissen FERTI*
- *Lesen der Treffersätze LES*

In der Durchrechnung der Arbeitslasten werden nicht die Parameter aller angegebenen Programme aufbereitet, sondern das Ergebnis einer in SEQUEL definierten Anfrage in Form von Angaben wie Zahl der Suchmoduln, Suchlaufsequenzen, Suchmodulassembler und Zwischenergebnisverarbeitung gemacht.

Aus der Anfrage und der Suchstrategie ergibt sich die Anzahl AES der Suchläufe für den Aufbau der Trefferlisten und die zugehörigen Zeiten:

1. Trefferlisten
1.1 Positionieren auf Dateianfang ca. ZSZ x AES
1.2 Leseumdrehungszeit AZD x HUP x 2 x AES
1.3 Zylinderfolgezeiten (AZD - 1) x ZFZ x AES

Das Lesen der einzelnen Datensätze erfordert direkte Zugriffe entsprechend dem Zugriffsmodell DAM in 3.2.4.1.2:

2. Lesezeiten für einen Datensatz
2.1 Positionieren Zylinder in Datei AZD x STi/3 + AWi
2.2 Positionieren in der Spur HUP
2.3 Lesezeit für Datenblock PSL/BMS
2.4 Das Isolieren der Datensätze aus Datenblöcken aufgrund der Byteadresse wird vernachlässigt. Die Isolierung einzelner Datenelemente wird nicht betrachtet. Insgesamt wird eine Zeitkorrektur durch 1ms angenommener Suchrechnerverwaltungszeit und 1,6ms KORRFAK wie bei SAM oder DAM berücksichtigt.

Für nach Zylinderreihenfolge vorsortierte Zugriffe gilt bei einer Trefferdichte von ATZ = Anzahl Treffer pro Zylinder die abgewandelte Form, falls bereits auf Dateianfang positioniert war:

2.1 STi/ATZ + AWi, für alle ATZ $<$ 1
2.1 ZFZ x N, alle ATZ $\geq$ 1 und N = (Anzahl Treffer)/ATZ.

Bei extrem hoher Trefferrate werden nur noch Spuren markiert, so daß sequentielles Lesen nach dem Zugriffsmodell SAM in 3.2.4.1.1 erforderlich wird. Über die Selektivität (98) läßt sich ein Erwartungswert für die Trefferzahl pro Block angeben, so daß der Suchrechner bei Durcharbeitung der Trefferliste mehrere Treffer pro Block abarbeiten und dadurch Lesezeiten verkürzen könnte. Bei sehr hohem Auflösungsfaktor (98) eines Attributes ist mit DAM-Verhalten der Ergebnisblöcke zu rechnen, bei einem niedrigen Auflösungsfaktor mit SAM-Verhalten.

Die Tabelle 3.7 (Zugriffszeiten in den Relationen) enthält die Anzahl der Zylinder, die Zeit für die Treffervektorerstellung bei AES = 1, den DAM Datensatzzugriff mit Get Unique und Get Next Lesezeiten für einen Satz und die gesamte Relation nach dem Zugriff SAM (Zeiten in ms).

RELATION	AZD	AES=1	GET UNIQUE	GET NEXT	TOTAL
IDENT	58	1578	26,2	1,94	234224
KLINIK	51	1391	24,8	2,31	280764
LABOR	200	5370	32,7	2,09	1688031
DIAG	144	3875	31,9	2,23	560915

TABELLE 3.7

Suchrechner: Zugriffszeiten in den Relationen *(in ms)*

3.2.4.2 Arbeitslasten und Ermittlung der Antwortzeiten

3.2.4.2.1 Datenbestände für den Vergleich

Die Auswahl der Arbeitslasten erfolgte aus dem Spektrum der Aufgaben des Bereiches der Datenbankgruppe des Medizinischen Systems Hannover. Die Erstellung von Datenstrukturen für den Suchrechner trägt diesen Arbeitslasten Rechnung. Eine 1 : 1 Übersetzung der Hierarchien des MSH ist zwar technisch einfacher (auch 3.2.4.1), bringt aber nicht die den Arbeitslasten entsprechende suchrechnergerechte Datenstruktur. Deshalb wurde ein Kompromiß gewählt, um die Datenmenge zu reduzieren (Begrenzung des Suchrechners auf eine Platteneinheit) und dennoch eine ausreichend komplexe und damit vergleichbare Datenstruktur (fünf Relationen mit gegenseitigen Beziehungen) zugrunde zu legen. Die betroffenen Hierarchien sind in den Bildern 3.7 bis 3.10 mit den Segmenttypen und Attributnamen vorgestellt. Die Beziehung Arbeitslast, Hierarchie (Segmenttypen), Relation (Relationsnamen) ist in Bild 3.11

gezeigt. Die Attributnamen sind weitgehend selbsterklärend gewählt. Die Attribute der Segmenttypen AUROOT, DIAGINVERT, NAROOT und PATSTAT (MSH, invertierte Dateien) können im Suchrechner entfallen. Die Einsparung dieser Bytemengen wurde beim Vergleich des erforderlichen Speicherplatzes nicht berücksichtigt, da die Zahl der Invertierungen nur von den Erfordernissen der jeweiligen Rechneranwendungen in der Praxis abhängt und nicht charakteristisch ist. Dennoch kann davon ausgegangen werden, daß zusätzliche Invertierungen in jedem nicht relationalen DB-System erforderlich sind, so daß der Vergleich des Speicheraufwandes letztlich günstiger für den Datenbankrechner ausfällt, als es der 1 : 1-Vergleich bei der Umsetzung hierarchischer Pfade in Tabelle 3.8 ausdrückt:

DATENBANK	LSL	PFXS	ALS	NAME	LNG	ANZAHL	GP/DB
CONPAT	313	76	120575	COROOT	117	120575	1,39
				COORT	18	241400	
				COSTA	8	4935	
				COGEB	15	39282	
				COBEHA	35	172031	
				COQUAL	40	70661	
				CODIKO	23	251497	
APAT	781	220	65315	APROOT	110	65315	1,33
				APBE	50	91740	
				APARZT	76	269681	
				APLDAT	10	67457	
				APWERT	20	808199	
LEIPAT	642	83	62990	LEROOT	64	62900	1,2
				LEAUFN	26	87165	
				LEKOST	106	91150	
				LEKLI	44	121737	
				LESOZI	104	171889	
NAMPAT	33	9	118857	NAROOT	27	118857	1,26
				NAREST	50	15788	
AUFPAT	18	9	139144	AUROOT	18	139144	1,5
PATSY	100	-	120575	PATSY	100	120575	1,0

TABELLE 3.8

Speicherplatzfaktor bei 1 : 1 Umsetzung von hierarchischen Pfaden in Relationen. Nicht berücksichtigt ist die Fähigkeit des Suchrechners zur Kategorienbildung und die im hierarchischen System erforderlichen zusätzlichen Invertierungen. Die Abkürzungen LSL, PFXS usw. entsprechen denen der Zugriffszeitmodelle. GP/DB soll Mehrzweckrechner zu Datenbankrechner (GPC/DBC) für Speicherplatzbedarf charakterisieren.

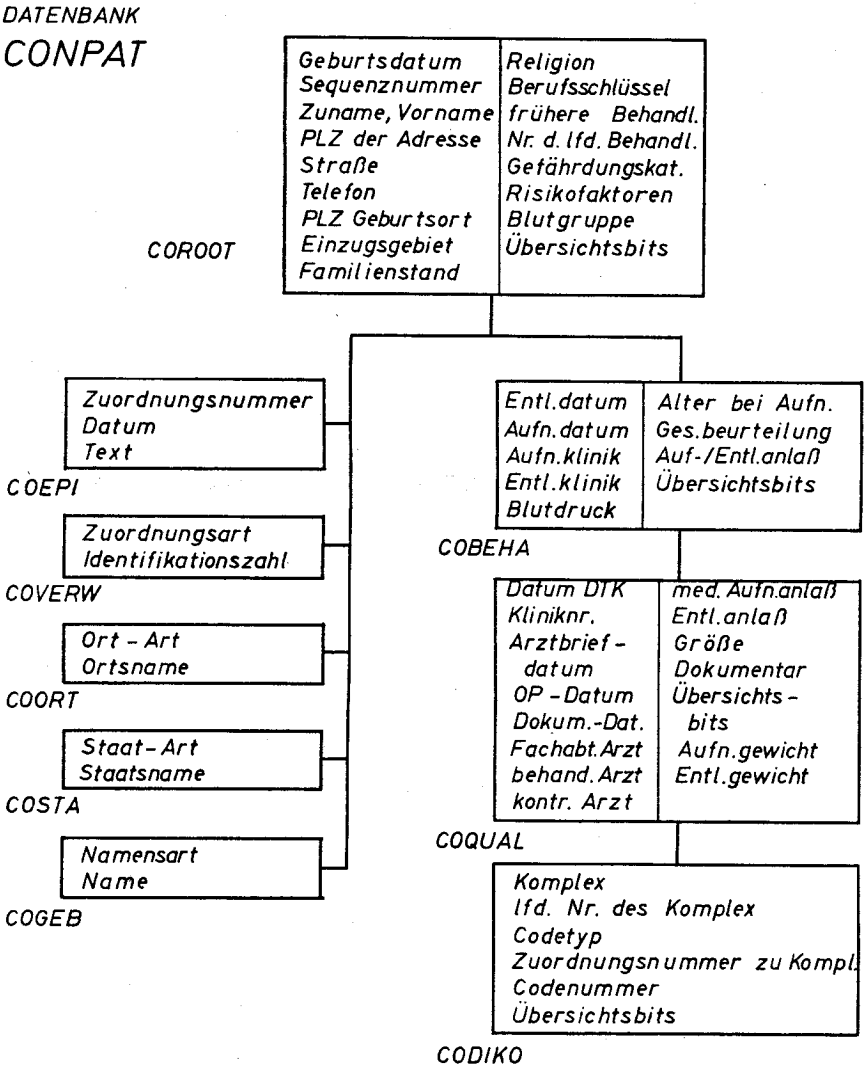


BILD 3.7

Hierarchische Segmentstruktur (Datensatzhierarchie) der Datenbank CONPAT mit den wichtigsten Datenelementen.

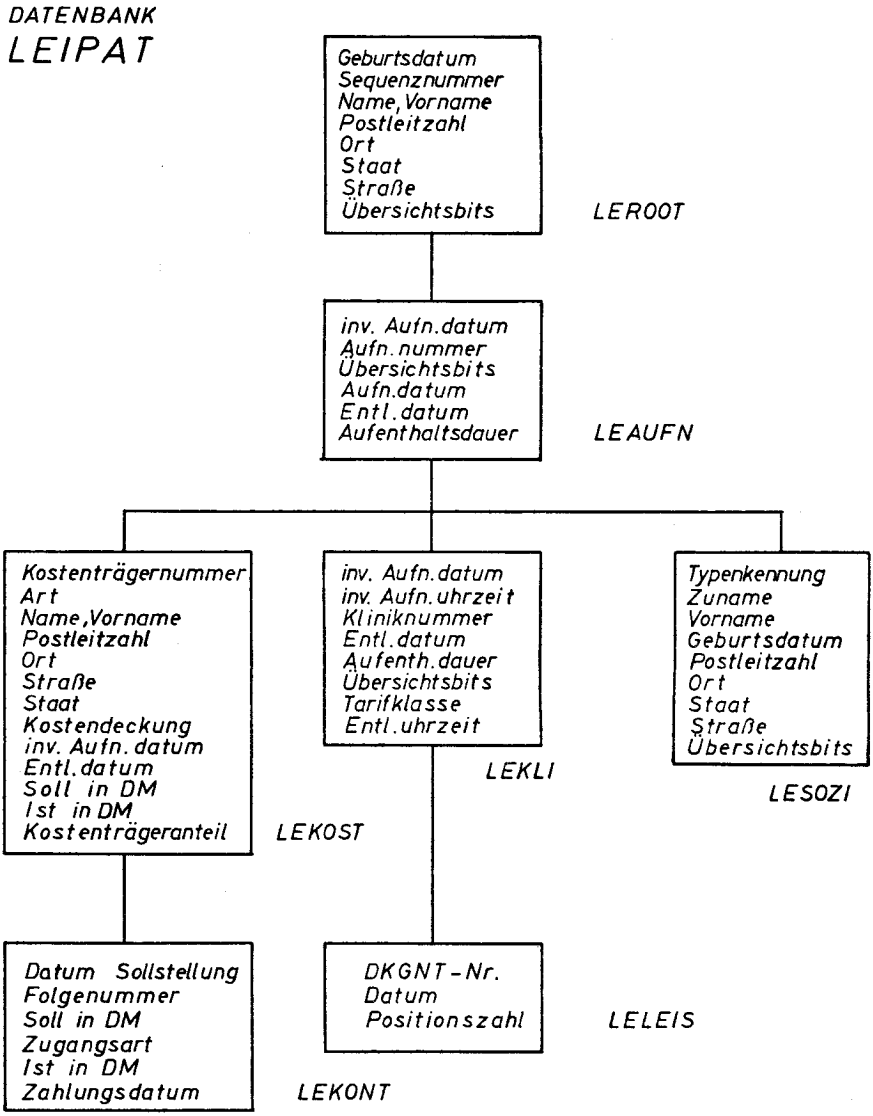


BILD 3.8

Hierarchische Segmentstruktur der Datenbank LEIPAT mit den wichtigsten Datenelementen.

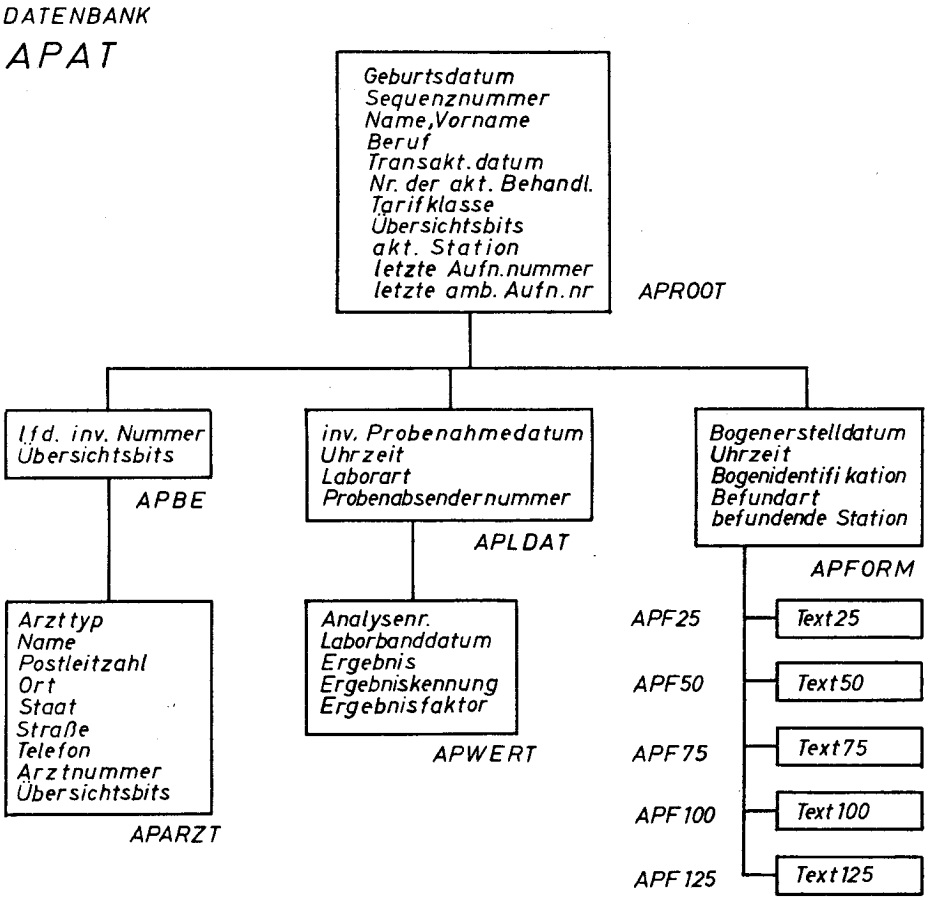


BILD 3.9

Hierarchische Segmentstruktur der Datenbank APAT mit den wichtigsten Datenelementen.

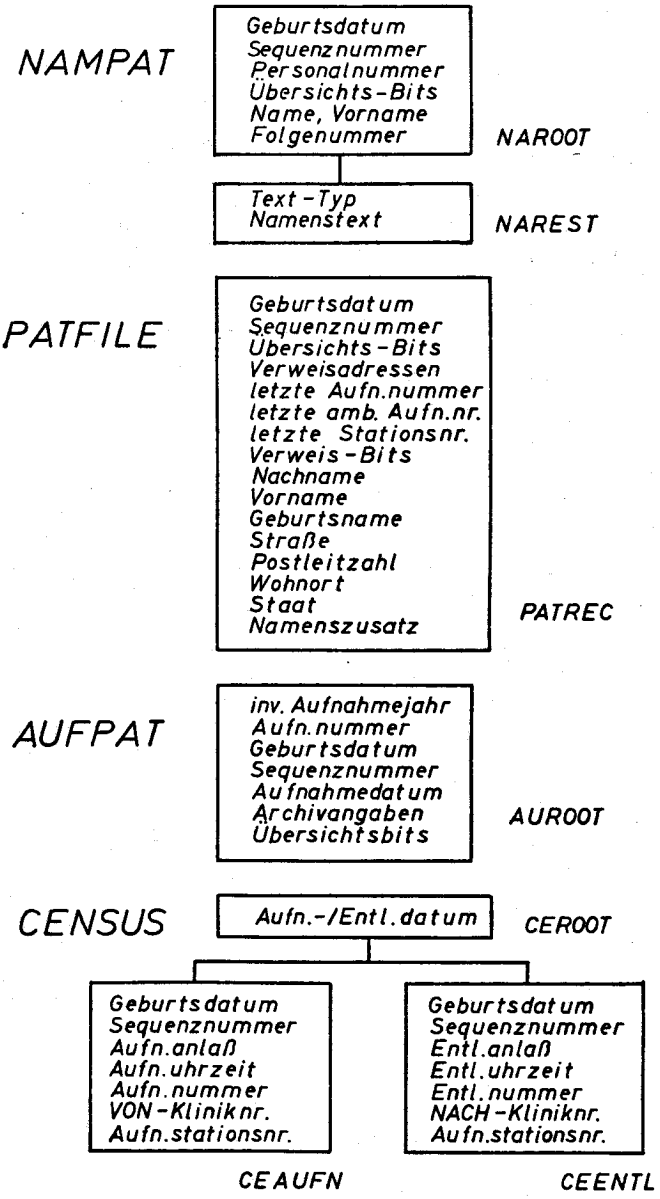


BILD 3.10

Hierarchische Segmentstruktur der Datenbanken NAMPAT, AUFPAT, CENSUS und PATFILE mit ihren wichtigsten Datenelementen.

Arbeitslast	Datenelemente aus Datensätzen																			Relationen				
	PATREC	COROOT	COBEHA	COQUAL	CODIKO	DIAGINV	APROOT	APLDAT	APWERT	AUROOT	LEROOT	LEAUFN	LEKLI	CEROOT	CEAUFN	CEENTL	NAROOT	LABORTEXT	PATSTATION	IDENT	KLINIK	LABOR	DIAG	LABTEXT
3.2.4.2.3.1							+							+	+	+				+	+			
3.2.4.2.3.2	+																			+				
3.2.4.2.3.3													+								+			
3.2.4.2.3.4											+		+				+			+	+			
3.2.4.2.3.5	+									+										+	+			
3.2.4.2.3.6	+																			+				
3.2.4.2.4.1							+	+	+									+				+		+
3.2.4.2.4.2		+																		+				
3.2.4.2.4.3		+	+																		+			
3.2.4.2.4.5							+				+	+	+								+			
3.2.4.2.4.6		+			+																		+	
3.2.4.2.4.7											+		+								+			
3.2.4.2.5.1		+	+	+	+	(+)																	+	
3.2.4.2.5.2		+	+	+	+	(+)																	(+)	
3.2.4.2.6	+													+		+				+	+			
3.2.4.2.7							+	+	+													+		
3.2.4.2.8.1		+	+																		+			+
3.2.4.2.8.2											+		+							+	+			
3.2.4.2.8.3		+									+									+	+			
3.2.4.2.8.4					+	(+)																	+	
3.2.4.2.8.5												+	+								+			
3.2.4.2.8.6													+								(+)			
3.2.4.2.8.7					+	(+)	+		+													+	+	
3.2.4.2.8.8		+	+	+	+																		+	
3.2.4.2.8.9				+	+																		+	
3.2.4.2.8.10			+	+	+																		+	
3.2.4.2.4.4											+		+								+			

BILD 3.11

Darstellung, bei welcher Arbeitslast aus welchen Datensätzen (Segmenten) bzw. den äquivalenten Relationen des Suchrechners Datenelementwerte zu verarbeiten sind. Die Klammern beim Datensatz DIAGINV deuten darauf hin, daß die Diagnoseinvertierung hier besser genommen würde. Die Klammern bei den Relationen deuten auf Aufgaben, die der Suchrechner nicht lösen kann, weil abgelei-tete Attributwerte benötigt werden.

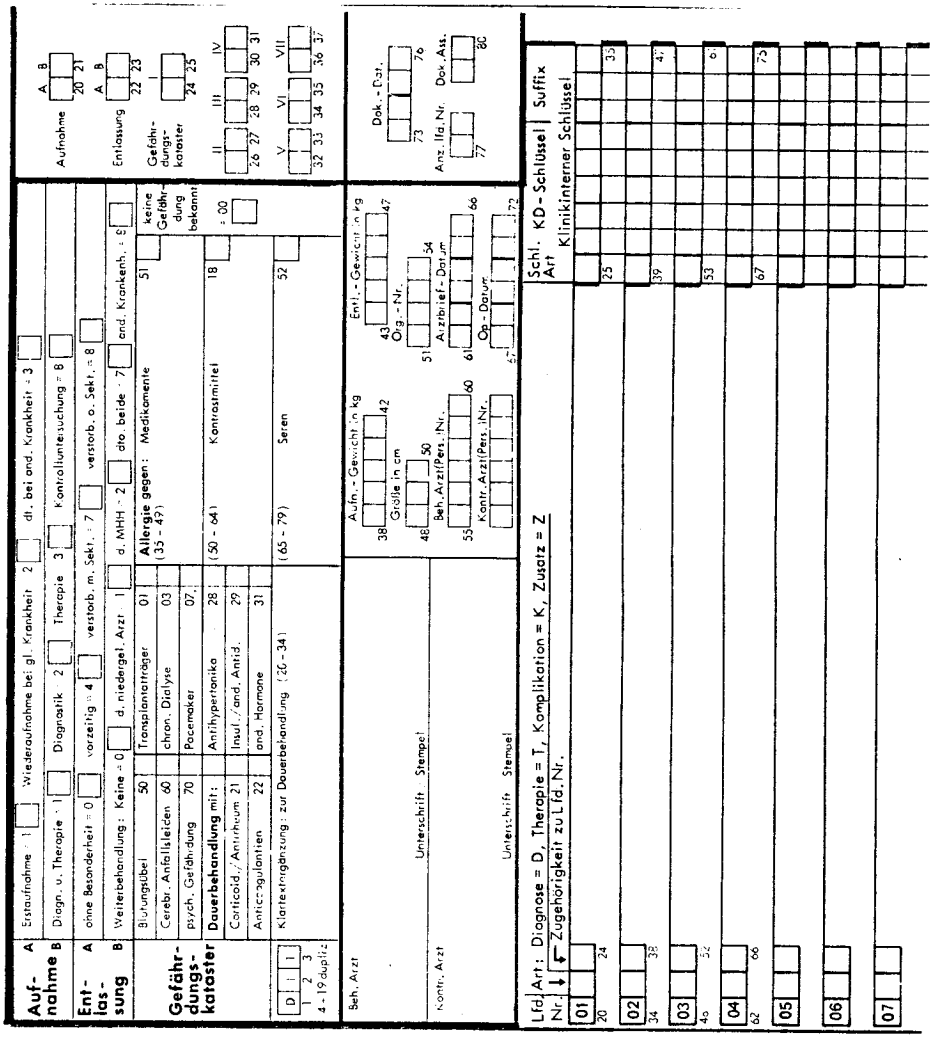


BILD 3.12

Formblatt für die Basisdokumentation der Krankenakte an der Medizinischen Hochschule Hannover zur Erläuterung der Zuordnungssystematik von Diagnosen über laufende und Zuordnungsnummern. Diese Nummern kehren auch in den Tupeln der Relation DIAG wieder.

Es wäre praktisch, wenn alle Laborwerte zu einem Probenahmedatum als Datenelementvektor darstellbar wären. Dies entspräche der zweistufigen Hierarchie im Sinne des Suchrechners, die gleichbedeutend mit der Wiederholungsgruppe (repeating group) (98) oder einem ARRAY (*) in PL/1-Notation ist. Eine Wiederholungsgruppe ermöglicht, die variable Anzahl von Werten eines Datenelementes (Attributes) in einem Datensatz zu speichern. So könnte unter dem Attribut "Beruf des Patienten" nicht nur ein Wert, sondern mehrere Berufsangaben desselben Patienten festgehalten werden. In normalisierten Relationen sind mehrere Datensätze (Tupel) dafür erforderlich, im Suchrechner jedoch nicht. Bei den Daten des MSH sind aber fast nie reine Attributrelationen, sondern ganze Gruppen voneinander abhängiger Attributwerte als Vektor zu sehen. Während Vornamen eines Patienten eine ideale Attributrelation darstellen, weil jeder Wert nicht von anderen Attributwerten abhängig ist, liegt bei Diagnosen oder Laborwerten eine Abhängigkeit von weiteren Attributwerten vor, so daß diese ebenfalls in den Vektor übernommen werden müssen. Man kann die Kette all dieser Attributwerte als einen neuen Attributwert auffassen, muß dann aber z.B. Bedingungen zu Einzelattributen als Teil (Maskierung) eines "Superattributes" ansehen. Dadurch wird die Programmierung der SURE-Aufträge und vor allem die Datenbeschreibung des Systems für die automatische Übersetzung relationaler Suchanfragen in SURE-Aufträge unnötig kompliziert. Es zeigt sich, daß die auf den ersten Blick so verlockende Zweistufigkeit des SURE-Modells für praktische Anwendungen aus der MSH-Datenbank keine Erleichterung bringt, sondern zusätzliche Komplikationen.

Die Unterteilung von Sätzen in Felder einer Kategorie erfolgt durch das Sonderzeichen "F". Das Umschaltzeichen "U" wird nicht benutzt, weil die Daten in den Suchrechnerrelationen sich im externen Format (CHARACTER) befinden sollen. Das Kategorienende "K" ist gleichzeitig Satzende "S", so daß sich ein Suchrechnersatz aus n Feldern wie folgt aufbaut:

"F" FELD1 "F" FELD2 "F" ... "F" FELDn-1 "F" FELDn "K" "S"

Die Relationen wurden entgegen der anfänglichen Zielsetzung nicht erstellt und auf den Suchrechner übertragen. Es reichte aus, Zeitmessungen und Modelle auf dem einen System im MSH durchzuführen, da es plausibel ist, gleiche Modelle für die Suchrechnerzeiten anzunehmen (siehe Kap. 3.2.4.1.7).

Die Relationen werden im folgenden durch ihre Attributnamen, -längen und die Angabe des Mengengerüsts beschrieben:

IDENT RELATION		
LÄNGE	I	ATTRIBUT
13	I	Patientenidentifikationszahl (TT.MM.JJ SEQU)
20	I	Beruf
12	I	Name
10	I	Vorname
10	I	Namenszusatz
12	I	Straße
12	I	Ort
3	I	Staat
2	I	Geburtsjahrhundert
10	I	Trennzeichen "F", "K", "S"
104	I	9 Attribute

TABELLE 3.9

Attributnamensliste und Feldlängen für die Relation IDENT.

104 x 120575 = 12539800 Byte in der realen Relation IDENT
1280 Blocklänge, 9 Blöcke/Spur 3330
120575 x 104/1280 = 9797 Blöcke
9797/9 = 1089 Spuren
1089/19= 58 Zylinder

KLINIK RELATION		
LÄNGE	I	ATTRIBUTNAME
13	I	Patientenidentifikationszahl
8	I	Aufnahmedatum Behandlung
5	I	Aufnahmenummer
8	I	Entlassungsdatum Behandlung
4	I	Aufnahmedauer in Tagen
8	I	Aufnahmedatum auf Station
2	I	Aufnahmeuhrzeit auf Station
4	I	Stationsnummer
8	I	Entlassungsdatum von Station
3	I	Tarifklasse
2	I	Entlassungsuhrzeit von Station
1	I	Aufnahmeanlaß Station
1	I	Entlassungsanlaß von Station
2	I	Kostenträgercode
15	I	Trennsonderzeichen "F", "S", "K"
84	I	14 Attribute

TABELLE 3.10

Attributnamensliste und Feldlängen für die Relation KLINIK.

84 x 121737 = 10225908 Byte in der Relation KLINIK
768 Blocklänge, 14 Blöcke/Spur 3330
10225908/768/14 = 952 Spuren
952/19 = 51 Zylinder

LABOR RELATION	
LÄNGE	ATTRIBUTNAME
13	Patientenidentifikation
8	Probenahmedatum
4	Uhrzeit
4	Absendestelle
3	Analysenummer
5	Ergebnisgrundwert
1	Kennung (pathologisch)
1	Ergebnisfaktor für Präsentation und Rechnung
9	Trennsonderzeichen "F", "K", "S"
48	8 Attribute

TABELLE 3.11

Attributnamenliste und Feldlängen für die Relation LABOR.

DIAG RELATION	
LÄNGE	ATTRIBUTNAME
13	Patientenidentifikationszahl
8	Entlassungsdatum, Behandlung
8	Aufnahmedatum, Behandlung
4	Aufnahmekliniknummer
4	Entlassungskliniknummer
2	Blutdruck
2	Alter bei Aufnahme in Jahren
1	Aufnahmeanlaß
1	Entlassungsanlaß
8	Datum der Diagnosestellung
4	Diagnoseklinik
8	Arztbriefdatum
8	Dokumentationsdatum
4	Fachabteilung
1	Medizinischer Aufnahmeanlaß
1	Medizinischer Entlassungsanlaß
3	Dokumentationsassistentin
3	Aufnahmegewicht
3	Entlassungsgewicht
2	Laufende Nummer
1	Codetyp (Diagnose, Therapie oder Komplikation)
2	Zuordnungsnummer zu laufender Nummer
9	Diagnosecode mit sog. Suffix
25	Trennsonderzeichen "F", "K", "S"
125	23 Attribute

TABELLE 3.12

Attributnamenliste und Feldlängen für die Relation DIAG.

125 x 251497 = 31437125 Byte in der Relation DIAG
1280 Blocklänge, 9 pro Spur 3330
(31437125/1280)/9 = 2729 Spuren
2729/19 = 144 Zylinder

LABTEXT	RELATION
LÄNGE	ATTRIBUTNAME
3	Laufende Nummer
20	Laboranalysebezeichnung
3	Trennsonderzeichen "F", "K", "S"
26	2 Attribute

TABELLE 3.13

Attributnamenliste und Feldlängen für die Relation LABTEXT.

26 x 512 = 13312 Byte in der Relation LABTEXT
520 Blocklänge, 20 pro Spur
2 Spuren 3330

Die Relationen für die Zeitvergleiche haben von der Byteanzahl her zu den hierarchischen Pfaden , aus denen sie abgeleitet sind, etwa ein Verhältnis 1,1 : 1. Das ergibt bei den Zeitvergleichen einen geringen Nachteil für die Relationen und damit den Suchrechner, vergl. Tabelle 3.8.

RELATION	SEGMENTE	DBC/GPC
IDENT	PATSY, (COORT)	1,13
KLINIK	Izahl, LEAUFN, LEKLI, (LEKOST)	1,14
LABOR	Izahl, APLDAT, APWERT	1,125
DIAG	Izahl, COBEHA, COQUAL, CODIKO	1,056

TABELLE 3.14

Speicherplatzverhältnis der im Vergleich der Zugriffszeiten benutzten Relationen zu den entsprechenden hierarchischen Pfaden.

3.2.4.2.2 Trefferhäufigkeit und Komplexität der Anfrage

In Modellen der Zugriffszeitschätzung spielt die Anzahl der zu erwartenden Treffer eine wichtige Rolle. Sie entscheidet, ob die sequentielle Verarbeitung schneller ist, als die Ausnutzung von Zugriffswegen z.B. in Form von Invertierungen. Diesem Zentralthema ist der Teil II von Wedekind und Härder (98)gewidmet. Die Trefferhäufigkeit in den durchgerechneten Beispielen dieser Arbeit spielt eine Rolle bei der Quantifizierung des Zugriffsaufwandes. Die Trefferhäufigkeit wird bei Wedekind

über den Auflösungsfaktor eines Attributes definiert:

Es seien j unterschiedliche Attributwerte für ein Attribut vorhanden. Der Auflösungsfaktor eines Attributes ist definiert als R = Nrec/j, wobei Nrec die Anzahl der Datensätze ist, in denen das Attribut enthalten ist. Der Wert von R ist dann der Erwartungswert für die Anzahl der Sätze, in denen ein bestimmter Attributwert eines Attributes vorkommt, wenn alle Attributwerte mit gleicher Häufigkeit Ni auftreten:

$$N_i = R = N_{rec} / j \qquad mit \quad \sum_{i=1}^{j} N_i = N_{rec}$$

Für Schätzwerte ist die Annahme der Gleichverteilung gerechtfertigt, bei Genauigkeitsanforderungen jedoch müssen die Auftretehäufigkeiten der Attributwerte bekannt sein, so daß das zum Attributwert zugehörige R genau ist. So ist eine typische Werteverteilung, die durch manuelle Auswahl aus einem vorgegebenen Spektrum erfolgt (Laboranalysen oder Diagnosen), extrem ungleich verteilt.

Enthält eine Anfrage die Bedingung WHERE (Attributwert = "Wert"), so läßt sich hieraus die Anzahl der Treffer mit AZT = THF x CARD (Relation) abschätzen, wenn die Trefferhäufigkeit THF = 1/j und CARD (Relation) die Mächtigkeit (oder Anzahl Datensätze) der zu überprüfenden Relation ist.

Eine Anfrage kann mehrere Bedingungen (Aussagen) in disjunkter Verknüpfung enthalten:

WHERE ($A_1 = a_1$ "ODER" $A_2 = a_2$ "ODER" ... "ODER" $A_n = a_n$),
mit A_i sind Attributnamen, a_i sind Attributwerte, $A_i = A_j$ ist erlaubt für $i, j \leq n$.

Die Trefferhäufigkeiten addieren sich, sofern die Auftretewahrscheinlichkeiten voneinander unabhängig sind. Tatsächlich aufzusuchen sind wenigerTreffersätze, da durch die gegebenen Verbundwahrscheinlichkeiten auch Treffer zusammenfallen werden.

$$AZT = \left(\sum_{1}^{n} THF_i \right) \cdot CARD(Relation)$$

Eine Anfrage kann mehrere Konjunktionen von Bedingungen enthalten:

WHERE ($A_1 = a_1$ "UND" $A_2 = a_2$ "UND" ... "UND" $A_n = a_n$),
mit $A_i \neq A_j$ für $i \neq j$.

Die voneinander unabhängigen Trefferhäufigkeiten multiplizieren sich:

$$AZT = \left(\prod_{1}^{n} THF_i \right) \cdot CARD(Relation)$$

Abfragen mit Bereichsangaben wie "größer" oder "kleiner" können für den Zweck der Trefferabschätzung ebenfalls in elementare Aussagen

mit dem Operator "=" umgewandelt (oder beliebig angenähert) werden, wenn die Werte in Klassen geteilt und jeder Klasse ein Auflösungsfaktor zugeordnet wird. So kann die Abfrage auf Alter "größer" 30 "und" Alter "kleiner" 55 aufgelöst werden z.B. in (Alter/10 = 3 "ODER" Alter/10 = 4 "ODER" Alter/10 = 5), wobei die Klassen prinzipiell beliebig fein gestuft werden können.

Die in Anfragen erlaubte Aussage

... Ai≠ai ...

führt für sich auf eine Trefferanzahl von AZT = (1 - THF) x CARD (Relation), so daß in den Produkten bzw. Summen der Trefferhäufigkeiten die Differenz zu 1 einzusetzen ist.

Bei üblichen Anfragen mit Klammerung findet formal eine Auflösung in Elementarformen statt, um über die Definition der Komplexität der Anfrage und den bekannten Auflösungsfaktoren der beteiligten Attribute eine Abschätzung der Treffer zu ermöglichen. Die Anfragekomplexität (98, 104) ist eine Funktion z.B. der mittleren Anzahl Konjunkte pro Disjunktion und der Anzahl der Disjunktionen einer Anfrage.

Mit der Abschätzung der Trefferzahl ist aber noch nicht ausgesagt, welcher Aufwand dahinter steht, diese Treffermenge zu gewinnen. Hier gibt es vor allem in relationalen Systemen Alternativen, die ganz entscheidende Zeitunterschiede allein aus der Reihenfolge von Mengenoperationen bedeuten. Auf die Optimierung dieser Zeiten durch die richtige Wahl der Mengenoperations- und Datenleseaktivitäten wird in dieser Arbeit nicht eingegangen. Der Weg des Findens der Treffer einer Anfrage wird nicht algorithmiert, sondern es wird versucht, diesen aufgrund der vorgegebenen Beispiele optimal anzunehmen. Dabei spielt also die Anfragekomplexität in allgemeiner Form, die auf die Mengenoperationen unmittelbaren Einfluß hätte, in diesem Sinn keine Rolle. Die Beispiele gehen von einer konstant vorgegebenen Anfragekomplexität aus (Ausnahme 3.2.4.2.7). Dennoch wird die Trefferhäufigkeit manchmal als Variable eingesetzt, so daß Zeitaussagen zwar für die Art der Anfrage konstant (speziell), jedoch bezüglich der Attributwerteigenschaften variabel (allgemein) sind.

3.2.4.2.3 Patientendatenauskunft

Die folgenden Beispiele entstammen der Patientenauswahl im Bildschirmauskunftssystem PIDS (100, 101). Die Antwortzeit der in PDPL ausgedrückten Arbeitslast bedeutet eine obere Zeitabschätzung gegenüber dem Programm PIDS. Nicht alle Beispiele werden in PDPL und SURE-Assembler dargestellt, weil die Art und Anzahl der Zugriffe eindeutig angebbar ist und die Notation gegenüber vorhergegangenen Beispielen nichts Neues bringt.

Das logische UND wird durch AND oder auch .AND. ausgedrückt. Der Stern, der im Suchmodul-Assembler Verwendung findet, ist durch ein + ersetzt. Größer als wird auch als .GT., kleiner als als .LT. geschrieben.

Die Definition der Aufgabe erfolgt in SEQUEL-Notation, in der der FROM-Teil auch weggelassen sein kann oder sich wahlweise auf Segmenttypen, Relationen oder Datenbanknamen bezieht. Die Punkte gliedern sich in Aufgabenstellung, PDPL-Zeitaufwand, SURE-Zeitaufwand und GPC/DBC-Zeitverhältnis. Die Namen Station und Klinik sind gleichwertig. Es wird unterschieden zwischen dem Gesamtaufenthalt der Behandlung (Beh) und den Einzelaufenthalten auf Stationen (Stat). Die Abkürzung Dat bedeutet Tagesdatum. Die Beispiele in PDPL- bzw. SURE-Notation sind nicht weiter erläutert, die jeweilige vollständige Syntaxdefinition ist in Kapitel 6 beigefügt.

3.2.4.2.3.1 Entlassungsliste / Aufnahmeliste

Aufgabe:

```
SELECT Izahl, Name
FROM Aproot
WHERE  Izahl =
  SELECT Izahl, Station, Anlass, Aufn_nr
FROM Census
  WHERE Entl_Dat_Stat='Datum' .AND. Station='   1'
```

PDPL:

```
DDC(Census, Izahl, Station, Anlass, Aufn_nr) DDC (Apat, Name)
DDC(Census, Entl_Dat_Stat, Station)
SFS(1,CEROOT) ACC(GU,1,=) CMD(NOCOUNT) CMD(COPY)
SSA()
  CPT(7='   1')  CPT(6='Datum')  ODB(SA,6)
CAL()  SCB(GE,MESS: Datum ohne Eintrag)
       SCB(XX,MESS: Systemfehler, Tel: 2670)
CMD(BCH,2)
SFS(2,CEENTL) ACC(GNP,O) CMD(NOPA)
SSA()
CAL() SCB(XX,BCH,1)
  IDB(IO,2,3,4)
  CMD(BCH,3)
SFS(3,APROOT) ACC(GU,1,=)
SSA() WHR(2=7)
      CMD(LGC,O,BCH,2)
      ODB(SA,2)
CAL() SCB(XX,MESS: Systemfehler, Tel: 2670)
      IDB(IO,5)  ODB(EX,1,2,3,4,5,6) CMD(BCH,2)
EOQ
```

Die Zugriffszeiten für ein Segment CEROOT (108ms) und angenommen 20 Segmente CEENTL (je 9,4ms) und 20 Segmente APROOT (je ca. 65ms) beträgt: 1596ms.

SURE:

```
SELECT  Izahl, Name FROM Ident WHERE Izahl =
SELECT  Izahl, Station, Anlass, Aufn_nr FROM Klinik
        WHERE Station='   1' .AND. Entl_Dat_Stat='Datum'
```

Zielliste erstellen für KLINIK in einem Suchmodul mit der Bedingung für das Datum und die Station in einem Suchlauf:

```
BEK +ENTL  TT.MM.JJ:
BNL: BNL: BNL:
BID +ENTL:
BNL: BNL: BNL: BNL:
BID '40404001':
BID +ENTL:
BKR,VTE:
BSR:
```

Die Zeit beträgt 1391ms. Die Zeit für das Einlesen der 20 Datensätze beträgt 496ms. Zu den 20 Identifikationszahlen sind jetzt die Datensätze in der Relation IDENT zu finden:

```
BID Izahl:
BKR,VTE:
BSR:
```

Bei AZT Treffern und ASM Suchmoduln ergeben sich AZT/ASM Suchläufe, daß bei 20 Izahlen und 13 Suchmoduln zwei Suchläufe erforderlich sind: 3156ms. Das Lesen der Datensätze benötigt noch 524ms, so daß in der Summe 5567ms vom Suchrechner benötigt werden.

GPC/DBC: 0,29

Die Aufnahmeliste ergibt dieselben Zeiten.

3.2.4.2.3.2 Geburtsdatumsliste

Aufgabe:

```
SELECT Izahl, Name, Geb_Name, Geb_Datum
  FROM Patsy WHERE Geb_Datum='Datum'
```

PDPL:

```
DDC(PATSY, Izahl, Name, Geb_Name, Geb_Datum, Sequ,
Geb_Datum)
SFS(1,Patsy) ACC(GU,1, >=)
  SSA()
   IDB(EX,4) oder CPT(4='Datum')
   CPT(5='0010')
   ODB(SA,4,5)
CAL() SCB(XX,MESS: Fehler, Tel: 2670)
   IDB(IO,1,2,3,6)
   WHR(6=4) CMD(LGC,O,MESS: Kein Patient gefunden)
   ODB(EX,1,2,3)
CMD(BCH,2)
```

```
SFS(2,Patsy) ACC(GN,O)
  SSA()
  CAL()
    SCB(XX,MESS: Zugriffsfehler) SCB(GB,RPUT)
    IDB(IO,1,2,3,6)
    WHR(6=4) CMD(LGC,O,RPUT)
    ODB(EX,1,2,3)
EOQ
```

Bei 35 Patienten sind die Einzelzeiten für einen Direktzugriff 46,7ms und 35 sequentiellen Zugriffen 138,3ms. Die Summe ist 185ms.

SURE:

Zielliste erstellen für IDENT in einem Suchlauf:
BID TT.MM.JJ/////: (/steht für das Markierungszeichen)
Die Treffervektorerstellung kostet 1578ms und das Lesen 917ms. Damit ist die gesamte Zeit 2495ms.

GPC/DBC: 0,074

3.2.4.2.3.3 Stationsliste

Aufgabe:

```
SELECT Izahl, Aufn_nr, Tarifkl, Name, Station
  FROM Lekli  WHERE Station='xyz'
```

PDPL:

Im MSH ist die stationsbezogene Invertierung PATSTAT nicht als Schnittstelle an GURS angeschlossen. Deshalb ist diese Abfrage im Gegensatz zu PIDS eine sequentiell in der Datenbank LEIPAT ablaufende Suche:

```
DDC(Leipat, Izahl, Aufn_nr, Tarifkl, Name, Station,
Entl_Dat_Stat)
SFS(1,Lekli) ACC(GN,O)
  SSA()
  CAL()
    SCB(XX,MESS: Zugriffsfehler) SCB(GB,RPUT)
    IDB(IO,1,2,3,4,5,6)
    WHR(5='xyz'.AND.6='O')  CMD(LGC,O,DMSC)
    ODB(EX,1,2,3,4,5)
EOQ
```

Die PDPL-Zeit ist 1490061ms.

SURE:

Die Trefferliste in der Relation Klinik kostet 1391ms.
BNL: BNL: BNL: BNL: BNL: BNL: BNL:
BID xyz:
BID '4040'_R:
BKR,VTE:
BSR:
Das Lesen der angenommenen 18 Treffer benötigt 446ms, so daß die Gesamtzeit 1837ms beträgt.

GPC/DBC: 811

3.2.4.2.3.4 (Teil-) Namenliste

Aufgabe:

SELECT Name, Izahl, Station
 WHERE Teilkette (Name) = 'Teilna-me'
Station ist hier die letzte Station, d.h. entweder Entlassungs- oder aktive Station.

PDPL:

```
DDC(Nampat, Name, Izahl, Aktiv, Name, Name, Name)
SFS(1,Naroot) ACC(GU,1,.GT.=)
 SSA()
  CPT(4='Teilna-me')
  CPT(6='TeilnameZ')
  ODB(SA,4)
CAL()
 SCB(xx,MESS:Fehler)SCB(GB,RPUT)
 IDB(IO,1,2,5)
CMD(BCH,2)
SFS(2,DUMMY)
  SSA()
  WHR(4 <=5 .AND. 5 <=6)
  CMD(LGC,0,RPUT)
  CAL()
   CMD(BCH,4)
SFS(3,NAROOT) ACC(GN,0)
 SSA()  CAL()
 SCB(xx,MESS:Fehler) SCB(GB,RPUT)
 IDB(IO,5)
 CMD(BCH,2)
SFS(4,Leroot) ACC(GU,1,=)
 SSA()
  CPT(3='+++')
  ODB(SA,2)
 CAL()
 SCB(XX,BCH,6)
 CMD(BCH,5)
```

```
SFS(5,LEKLI) ACC(GNP,0)
 SSA()
 CAL() SCB(XX,BCH,6)
  IDB(IO,3)
 CMD(BCH,6)
SFS(6,Dummy)
 CMD(ODB2)
 SSA()
 CAL()
  ODB(EX,1,2,3)
 CMD(BCH,3)
EOQ
```

Der Zeitbedarf bei 200 Treffern, von denen 10 aktiv (in Behandlung) sind: ein Direktzugriff Nampat mit 103,5ms, 200 mal sequentielle Zugriffe im Nampat mit 1616ms, 200 mal Direktzugriffe im Leipat mit 14100ms und 200 mal ein Folgesegment im Leipat mit Get Next Within Parents lesen, 2340ms.
Die Zeiten für die Dummy-Pfade, die keinen Datensatzzugriff beinhalten, gehen 400 mal mit je einer ms ein. Die Gesamtzeit ist 18559ms.

SURE:

```
IDENT Relation für die Namensbedingung
BNL: BNL:
BID Teilna-me/////////:
BKR,VTE:
BSR:
```

Die Zeit beträgt 1578ms, die Zeit für das Lesen von 200 Treffern 5240ms.

```
BEV +Datum 8_E
BID Izahl:
BNL:
BLV +Datum:
BNL: BNL: BNL: BNL: BNL:
Bild +Datum
BKR,VTE:
BSR:
```

Die jüngste Klinik wird aus der Relation KLINIK ermittelt. Dabei ist die Bedingung, daß Entlassungsdatum aus der Behandlung gleich mit dem Entlassungsdatum aus der Station ist. Zusätzlich wird die Izahl geprüft. Bei 13 Suchmoduln sind 16 Suchläufe je zu 1319ms und 200 Datensatzzugriffe je zu 24,8ms erforderlich. Die Gesamtzeit beträgt 34034ms.

GPC/DBC: 0,55

3.2.4.2.3.5 Aufnahmenummernliste

Aufgabe:

```
SELECT Aufn_nr, Aufn_Jahr, Aufn_Datum, Izahl
 WHERE Aufn_nr='n' .UND. Izahl=
   SELECT Izahl, Name FROM PATSY
```

PDPL:

Die Aufnahmenummer mit dem Aufnahmejahr bilden den Wurzelsegmentschlüssel in der Datenbank Aufpat. Das Jahr ist invertiert, so daß das größte Jahr physisch zuerst gefunden wird (1980=20, 1981=19, usw.).

```
DDC(Aufpat, Aufn_Jahr, Aufn_Nr, Aufn_Datum, Izahl)
DDC(Patsy, Izahl, Name)
SFS(1,Dummy)
 SSA()
  CPT(2='Aufn.nr') CPT(7='20')
 CMD(BCH,2)
 CAL()
SFS(2,Auroot) ACC(GU,1,=)
 SSA()
  CPT(7+'1')   ODB(SA,7,2)
  WHR(7.LT.'27')  CMD(LGC,0,RPUT)   Schleifenende
 CAL()
  SCB(XX,STOP)   SCB(GE,CPT1)
  IDB(IO,1,2,3,4)
 CMD(BCH,3)
SFS(3,Patsy) ACC(GU,1,=)
 SSA()
  CPT(6='+++') CPT(5=4)
  ODB(SA,5)
 CAL()  SCB(XX,ODB2)
  IDB(IO,6)
  ODB(EX,1,2,3,4,6)
 CMD(BCH,2)
EOQ
```

Die Zugriffszeiten für fünf Aufnahmejahre mit der Aufnahmenummer sind fünf Direktzugriffe im Aufpat mit 490ms und fünf Direktzugriffe im Patsy mit 234ms. Die Gesamtzeit ist 724ms.

SURE:

```
BEV +Datum 8_E:
BNL:
BLV +Datum
BNL: BNL: BNL:
BID +Datum
BID Aufnahmenummer
BKR,VTE:
BSR:
```

Die Zeit für die KLINIK Relation beträgt 1391ms (Treffervektorerstellung) und für das Lesen von 5 Datensätzen 124ms.
Die Suche in der Relation IDENT benötigt 1578ms und das Lesen der fünf Treffersätze 131ms, so daß die Gesamtzeit im Suchrechner mit 3224ms anzunehmen ist.

GPC/DBC: 0,22

3.2.4.2.3.6 Identifikationsdaten

Aufgabe:

```
SELECT Name, Izahl, Geb_Name, Straße, Wohnort
  WHERE Izahl='Izahl'
```

PDPL:

Ein Direktzugriff im Patsy mit 46,7ms.

SURE:

Relation IDENT, Suchen und Lesen in der Zeit von 1604ms.

GPC/DBC: 0,029

3.2.4.2.4 Patientendatenauskunft, Datenoptionen

3.2.4.2.4.1 Labordatenliste

Aufgabe:

SELECT Laborwert, Faktor, Pathol, Lab_Datum,
Lab_Analys_Name
WHERE Izahl='Izahl'

Bemerkung: Die externe Darstellung des Laborwertes ist LABORWERT*10**(FAKTOR-2) für alle Laboranalysen mit FAKTOR≠7; für FAKTOR=7 ist das Laborergebnis ein Text in Abhängigkeit vom Laborwert. Auf diese Spezialität soll nicht eingegangen werden.

PDPL:

```
DDC(Apat, Laborwert, Faktor, Pathol, Lab_Datum,
Lab_Analyse_ Name, Izahl, Lab_Analyse_Nr)
DDC(TR, Schlüssel, Text)
SFS(1,Aproot) ACC(GU,1,=)
 SSA()
  IDB(EX,6) ODB(SA,6)
 CAL() SCB(XX,MESS: Fehler, Tel 2670)
       SCB(GE,MESS: Patient unbekannt)
 CMD(BCH,2)
SFS(2,Apwert) ACC(GNP,0)
 SSA()
 CAL() SCB(XX,RPUT)
  IDB(IO,1,2,3,4,7)
 CMD(BCH,3)
SFS(3,TR 01) ACC(GU,1,=)
 SSA()
  CPT(5='+++')    CPT(8=7)
  ODB(SA,8)
 CAL()  SCB(XX,BCH,4)
  IDB(IO,9)  CPT(5=9)
 CMD(BCH,4)
SFS(4,DUMMY)
 CMD(ODB2)
 SSA()
 CAL()
  ODB(EX,5,1,2,3,4)
 CMD(BCH,2)
EOQ
```

Zeitaufwand PDPL: a) etwa 80 Laborwerte, bzw b) keine Laborwerte.
Ein Direktzugriff Apat-Wurzelsegment a) und b) 70,2ms,
81 mal, bzw 1 mal sequentieller Zugriff auf Segmenttyp Apwert mit a) 912ms und b) 11,7ms,
80 mal, bzw 0 mal Zugriff in der Tabelle TR mit a) 1600ms und b) 0ms,

80 mal ein Zugriffspfad ohne Datensatzzugriff mit a) 80ms
und b) 0ms.
Gesamtaufwand: a) 2698ms und b) 82 ms.

SURE:

Suchlauf in der Relation LABOR mit der Izahl in der Zeit von 5370ms. Lesen der 80 Treffersätze in 80·32,7=2616ms. Suchlauf für Laboranalysebezeichnungen in 16,7ms und Lesen der Texte in 1528ms.

```
SELECT ..., Lab_Analyse_nr, Lab_Analyse_Name, ...
   FROM Labor
    WHERE Izahl='izahl' AND Lab_Analyse_nr=
    SELECT Lab_Analyse_nr, Lab_Analyse_Name
       From Labtxt.
```

Gesamtzeit, in der die Zuordnung der gelesenen Bezeichnungen zu den Datentupeln aus LABOR nicht berücksichtigt ist, ergibt a) 9531ms und b) 5370ms.

GPC/DBC:

a) 0,28 Liste mit 80 Werten
b) 0,015 Fehlanzeige

3.2.4.2.4.2 Gefährdungskataster

Aufgabe:

```
SELECT  Gefahr1, Gefahr2, ...
 WHERE  Izahl='Izahl'
```

PDPL:

Direktzugriff Wurzelsegment Conpat: 118,8ms.

SURE:

Suchlauf und Datensatzzugriff in IDENT: 1604ms.

GPC/DBC: 0,074

3.2.4.2.4.3 Behandlungsübersicht

Aufgabe:

```
SELECT  Aufn_Nr, Aufn_Datum_Beh, Station(!), Entl_Datum_Beh,
Station(!)
 WHERE   Izahl='Izahl'
```

Bemerkung: Station(!) ist einmal Aufnahmestation, einmal Entlassungsstation. Das Problem hängt mit der Struktur der Relation KLINIK zusammen, weil hier die Aufnahmestation in einem anderen Tupel vorkommt, als die Entlassungsstation. Die Aufnahmestation ist dadurch definiert, daß das Datum der Aufnahme in die Station identisch ist mit dem Datum der Aufnahme der Behandlung, entsprechendes gilt für die Entlassungsstation. Die oben in SEQUEL definierte Aufgabe ist so nicht richtig formuliert: die Lösung besteht aus zwei Teilaufgaben, deren Ergebnisse durch einen Verbund vereinigt werden müssen.

PDPL:

Angenommen werden zwei Aufenthalte, Lesen des Segmentes Coroot im Direktzugriff mit 118,8ms und Lesen zweier Segmente sequentiell vom Typ Cobeha in 22ms. Die Gesamtzeit ist 141ms.

SURE:

Suche in der Relation KLINIK, wobei die Aufnahmestation und die Entlassungsstation getrennt ermittelt wird und ein Verbund über Izahl und Aufnahmenummer ein gemeinsames Tupel erzeugt, in dem Aufnahme und Entlassungsstation enthalten sind. Suchläufe in 2782ms, Lesen der Tupel (2 mal 2) in 99 ms ergibt eine Gesamtzeit von 2881ms. Der Aufwand für den Verbund (Join) ist vernachlässigt.

```
BEV +Datum 8_E:
BID Izahl_E:
BLV +Datum:
BNL: BNL: BNL:
BID +Datum:
BKR,VTE:
BSR:
```

GPC/DBC: 0,049

3.2.4.2.4.4 Verlegungsübersicht

Aufgabe:

Alle Verlegungen von der Aufnahmestation bis zur Entlassungs- oder aktiven Station in zeitlich geordneter Folge.
Werden die Verlegungen nur zu einer bestimmten Behandlung und nicht zu allen Behandlungen gesucht, so ist im WHERE-Teil noch die Bedingung (AND Aufn_Nr='Nr') hinzuzunehmen.

```
SELECT  Station, Aufn_Datum_Stat, Aufn_Uhr_Stat, Tarifkl,
        Entl_Datum_Stat, Entl_Uhr_Stat, Aufn_nr
 WHERE  Izahl='Izahl'
```

PDPL:

Lesen im Direktzugriff Segmenttyp Leroot in 70,5ms und Lesen der abhängigen Segmente des Typs Lekli in sequentieller Folge in 3·11,7=35,1ms, falls drei Stationsaufenthalte vorhanden sind. Die Gesamtzeit beträgt 106ms.

SURE:

Ein Suchlauf in der Relation KLINIK in 1391ms und Lesen der drei Treffer in 74,4ms ergibt eine Zeit von 1465ms.

GPC/DBC: 0,072

3.2.4.2.4.5 Administrativdaten

Aufgabe:

Die Daten von mehr als 35 Attributen entstammen den Datenbanken Conpat, Apat und Leipat. Für die Aufgabe hier sollen lediglich die Zugriffswege ähnlich umfangreich sein, die Zahl der Attribute wird eingeschränkt.

```
SELECT  Izahl, Name, Straße, Ort, Beruf,
        Tarifkl, Aufn_nr , Aufn_Datum_Beh, Klinik,
        Entl_Datum_Beh, Klinik
 WHERE  IzahT='Izahl  AND Aufn_Nr='NR'
```

Das Problem der relationalen Sprachdefinition besteht darin, daß das Ergebnis aus den Attributwerten verschiedener Tupel derselben Relation besteht. Die Tarifkl soll z.B. diejenige bei Aufnahme sein. Klinik soll einmal Aufnahme-, einmal Entlassungsklinik sein. Ein Attribut besitzt aus dem Kontext (hier Attributwerte anderer Attribute des Tupels) heraus unterschiedliche Bedeutung. Man kann sich dadurch behelfen, daß man die Anfrage aus zwei Teilanfragen zusammenführt. Das wird für den Suchrechner gezeigt (vergleichbar mit 3.2.4.2.4.3).

PDPL:

1. Direktzugriff auf Segmenttyp APROOT für den Beruf in 70,2ms.
2. Direktzugriff für den Segmenttyp Leroot in der Zeit von 70,5ms.
3. Qualifiziertes sequentielles Lesen des Segmenttyps Leaufn, bis die Aufnahmenummernbedingung erfüllt ist (angenommen zwei Segmente) in 2·11,1=22,2ms.
4. Sequentielles Lesen der dem Segment Leaufn zugeordneten Segmente Lekli (angenommen drei Segmente) in 3·11,7=35,1ms. Die Gesamtzeit ist 198ms.

SURE:

1. Suchlauf für zwei Treffervektoren
   - Suchmodul 1: Izahl .UND. Aufn_Nr .UND. Aufn_Datum_Beh=Aufn_Datum_Stat
   - Suchmodul 2: Izahl .UND. Aufn_Nr .UND. Entl_Datum_Beh=Entl_Datum_Stat
   - Zeit: 1391ms
2. Erstellen zweier Zwischenergebnisrelationen
   - Izahl, Auf-Nr, Aufn_Datum_Beh, Klinik, Tarifkl
   - Izahl, Auf_Nr, Entl_Datum_Beh, Klinik

   Zeitbedarf: 49,6ms
3. Umbenennen der Attributnamen in der zweiten Zwischenrelation z.B. Klinik in Entl_Klinik. Diese Zeit ist nur zu werten, wenn ein Eintrag auf einem Sekundärspeicher erfolgt. Hier soll angenommen werden, daß der Vorgang im Hauptspeicher abgewickelt wird.
4. Verbund der beiden Zwischenergebnisse über Izahl und Aufnahmenummer. Diese Zeit ist in dieser Aufgabe auch vernachlässigt (Hauptspeicher).
5. Suchlauf und Einlesen in der Relation IDENT für die restlichen Attribute in 1578+26,2=1604ms

Die Gesamtzeit ist 1391+49,6+1604=3045ms.

GPC/DBC: 0,065

3.2.4.2.4.6 Liste der Diagnosen, Therapien und Komplikationen

Aufgabe:

```
SELECT DTKTyp, Zuordnung, Diagnose, Diag_datum, Dia_Klinik, Arzt
 WHERE Izahl='Izahl'
```

PDPL:

Bei einer Zahl von 6 Diagnosen ergibt sich ein Direktzugriff auf das Segment Coroot in 118,8ms und sechs Zugriffe auf den Segmenttyp Codiko sequentiell in 6·14,8=88,8ms.
Die Gesamtzeit ist 208ms.

SURE:

Suchlauf in der Relation DIAG in 3875ms und Lesen der sechs Treffersätze in 6·31,9=191ms.
Die Gesamtzeit ist 4066ms.

GPC/DBC: 0,051

3.2.4.2.4.7 Letzte Station

Aufgabe:

Für den Pförtner, z.B. die letzte, d.h. aktive oder Entlassungsstation mit Aufnahme- und Entlassungsdatum.

```
SELECT  Station, Aufn_Datum_Stat, Entl_Datum_Stat
 WHERE  MAX(Aufn_Datum_Stat)=Aufn_Datum_Stat UND
        Izahl = ⌜Izahl⌝
        MAX soll die zeitlich letzte Aufnahme auf Station bestimmen.
```

PDPL:

Direktzugriff auf das Wurzelsegment Leroot in 70,5ms und auf das physisch zuerst folgende abhängige Segment Lekli in 11,7ms. Die Summe ist 82ms.

SURE:

Suche in KLINIK Relation in 1391ms und Lesen der Treffersätze (angenommen drei Sätze) in 3·24,8=74,4ms.
Gesamtzeit: 1465,4ms.
Im Hauptspeicher muß ein Programm das jüngste heraussuchen, da diese Aufgabe nicht im Suchmodul-Assembler definierbar ist.

GPC/DBC: 0,056

3.2.4.2.5 Auswerteprogramm PATWERT

Das Auswertungsprogramm greift auf ein Stammband zu, dessen Daten aus dem hierarchischen Pfad des Segmenttyps CODIKO stammen. Datenelemente sind:

COROOT: Geburtsdatum, Geschlecht, Izahl, Familienstand, Religion,
COBEHA: Aufnahmejahr, Aufnahme-technischer Tag des Jahres, Alter bei Aufnahme, Alter in Tagen bei Aufnahme, Aufenthaltsdauer, verstorben/lebend entlassen, Krankenhaus, Station
CODIKO: Diagnosecode

Diese Datenelemente dienen der Selektion und Klassifikation von Patientensubpopulationen für statistische Auswertungen. Aus Originaldaten abgeleitete Größen sind Aufnahmedauer als Differenz von Aufnahmedatum und Entlassungsdatum in Tagen, das Geschlecht aus der Identifikationszahl, der technische Tag und das Jahr der Aufnahme aus der internen Datumsrepräsentation. Zur Klassifikation für z.B. anschließende statistische Auswertung muß z.B. das Alter der Patienten in Klassen einteilbar sein. Eine andere Einteilung über den technischen Tag erlaubt saisonale Klassenbildungen.

Diese Anforderungen werden durch die PDPL sämtlich unmittelbar erfüllt:

- Datumsdifferenz über CPT (Datum1 <> Datum2)
- Geschlecht als Datenelementverarbeitungsregel automatisch
- Technischer Tag und Jahr über Datendefinitionen, die durch Bezug auf Teil-Bitkette der Grundvariablen Tagesdatum als ganz normale Datenelemente ansprechbar sind.
- Klassenbildung durch die Grundrechenarten oder sonstigen Funktionen des CPT-Befehls.

3.2.4.2.5.1 Originaldatenauswertung

Aufgabe:

SELECT Attributliste des hierarchischen Pfades zu Codiko
WHERE Bedingungen an Attributwerte

PDPL:

Lesen aller Codiko Segmente mit dem IMS Pathcall (=einschließlich hierarchisch übergeorgneter Segmente). GURS unterstützt die Pathcall Datensatzsuchen, indem es im I/O-Bereich entsprechende Datenelementverschiebungen berücksichtigt; davon merkt der Benutzer nichts. Standardverarbeitung ist mit Pathcall. Ausschalten läßt sich diese über CMD(NOPA). Die Lesezeit beträgt 3360000ms. Verarbeitet GURS ein Stammband, dann verringert sich die Zeit je nach Blockung auf ein Drittel der Zeit.

SURE:

Die Anzahl der Suchläufe ist abhängig von der Anfrage. Bei 13 Suchmoduln reicht im allgemeinen ein Suchdurchlauf für die Trefferlistenerstellung. Die Zeit beträgt 3875ms.

Ist die Trefferdichte in der Suchelektronik zu hoch (Obergang zur Spurmarkierung), muß sequentiell gelesen werden. In diesem Fall ist der GPC/DBC-Faktor 3360000/560915=6.
Sonst werden die Treffer (AZT) mit Direktzugriffen gelesen.
Die Lesezeit ist dann AZTx31,9ms.

GPC/DBC:

3360000/(AZTx31,9+3875)
Mit AZT=THFxCARD(DIAG) und THF=Trefferhäufigkeit läßt sich folgende Form definieren:
GPC/DBC = 0,419/(THF+0,000483)

Die bildliche Darstellung GPC/DBC=f(THF) ist in Bild 3.13 dargestellt.

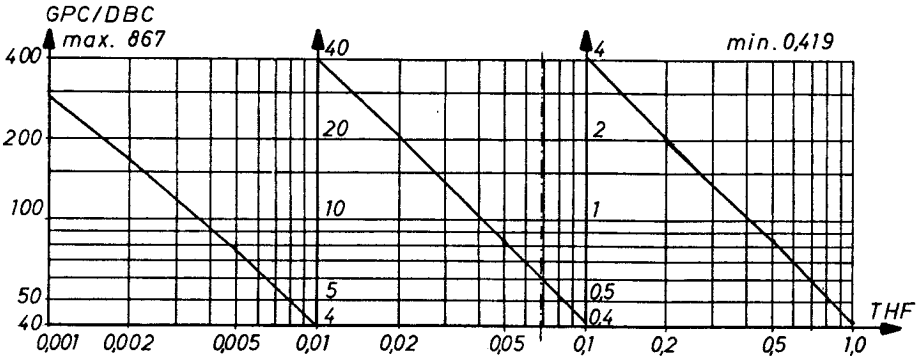


BILD 3.13

Antwortzeitverhältnis GPC/DBC als Funktion der Trefferhäufigkeit

Die Spanne reicht von THF=0, GPC/DBC=867 bis THF=1 und GPC/DBC=0,42.

Es läßt sich nun auch noch ein Vergleich angeben, der sich auf die sequentielle Verarbeitung im Suchrechner und die Verarbeitung mit der Suchelektronik im Suchrechner bezieht.: sequ. Lesen < = (Suchlauf + direktes Lesen).

560915 < = AZTx31,9+3875
THF=0,069 ist die Trefferhäufigkeit, ab der der Einsatz der Suchelektronik sich zeitlich nicht lohnt. Da sich die Trefferhäufigkeit über Komplexität der Anfrage und Auflösungsfaktoren der Attribute abschätzen läßt (3.2.4.2.2 , S. 88) ist die Eintragung dieser Grenze in das GPC/DBC Diagramm ebenfalls sinnvoll, wenn es für die Entscheidung, Suchrechner konventionell Lesen oder Suchrechner mit Suchelektronik Lesen, benutzt werden soll. In jedem Fall ist dann zu prüfen, ob wegen der parallelen Verarbeitung weiterer Suchanfragen diese Grenze im realen Betrieb des Suchrechners sinnvoll ist.

3.2.4.2.5.2 Auswertung abgeleiteter Attributwerte

Aufgabe:

SELECT Attribute des hierarchischen Pfades Codiko
und arithmetische und andere Ausdrücke dieser Attribute
WHERE Bedingungen zu den Attributen oder Ergebnissen der Verarbeitung der Attributwerte im SELECT Teil der Anfrage.

PDPL:

Wie bei Originaldatenauswertung, 3360000ms

SURE:

Die Anfrage mit z.B. ...WHERE Aufn_Dauer > 3 .UND. weitere ...
muß nicht auf das sequentielle Lesen der gesamten Relation führen, da die durch Konjunktion verbundene Aussage für weitere Bedingungen vorab auswertbar ist. Die zutreffenden Sätze sind zu lesen und in eine Zwischenergebnisrelation zu stellen, die dann sequentiell abzuarbeiten ist. Wenn dafür ein Programm zu schreiben ist, entzieht sich der GPC/DBC-Faktor der Abschätzung. Sonst läßt sich über die Trefferhäufigkeit die Größe der Zwischendatei schätzen, wie auch deren Verarbeitungszeit.

Die folgende Aussage gilt nur für eine Zwischendatei, nicht für mehrere Zwischendateien, die dann nötig werden, wenn disjunkt verbundene Bedingungen zu einem Attribut vorliegen.
Suchrechner Suchlauf: 3875ms.
Lesen der Treffersätze: AZTx31,9ms
Einstellen in eine Zwischenergebnisrelation auf dem Sekundärspeicher: ca. AZTx20ms.
Sequentielles Lesen mit einem Sonderprogramm, je nach physischer Speicherung der Zwischenergebnisrelation ca. AZTx5ms.
Damit ergibt sich ein Faktor von

GPC/DBC: 0,235/(THF+0,00027)

Diese Abhängigkeit ist im Bild 3.14 dargestellt, wobei die gestrichelte Linie die GPC/DBC-Funktion für nur Originaldaten aus Bild 3.13 dagegenstellt.

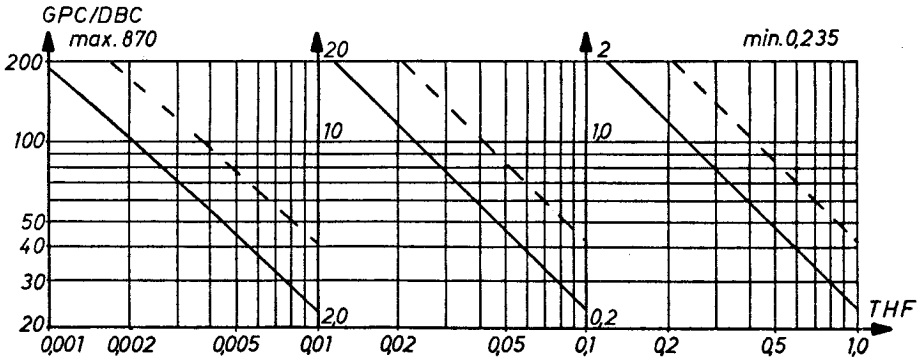


BILD 3.14

Antwortzeitverhältnis GPC/DBC als Funktion der Trefferhäufigkeit.

3.2.4.2.6 Patientenentlassungsliste

Aufgabe:

```
SELECT  Station, Entl_Art, Izahl, Name, Aufn_Datum_Stat,
        Entl_Datum_Stat, Aufn_Nr
 WHERE  Entl_Datum_Stat .NACH. 'TT.MM.JJ' .UND.
        Entl_Datum_Stat .VOR. 'TT.MM.JJ' .UND.
        Station='9999'
```

PDPL:

Da die interne Datumsrepräsentation eine Sortierung ermöglicht, kann GURS mit Direktzugriff in der Datenbank Census aufsetzen und dann sequentiell weitersuchen. Die Datumsgrenzen seien 26.3.78 und 1.4.78.

```
DDC(Census, Station, Entl_art, Izahl, Aufn_Datum, Aufn_Nr,
            Entl_Datum, Entl_Datum, Entl_Datum)
DDC (Patsy, Name)
SFS(1,Ceroot) ACC(Gu,1,=)
 SSA()
  CPT(7='26.03.78')   CPT(8='01.04.78')
  ODB(SA,8)
 CAL()   SCB(XX,STOP)
 CMD(BCH,2)
SFS(2,Ceroot)   ACC(GN,O)
 SSA()
 CAL()   SCB(XX,STOP)  SCB(GB,RPUT)
  IDB(10,6)
  WHR(6-=> 7.UND.6 =< -8)     CMD(LGC,O,RPUT)
CMD(BCH,3)
SFS(3,Ceentl)   ACC(GNP,O)
 SSA()
 CAL()   SCB(XX,BCH,2)
  IDB(IO,1,2,3,4,5,6,7)
  WHR(1='Station')   CMD(LGC,O,DMSC)
 CMD(BCH,4)
SFS(4,Patsy)   ACC(GU,1,=)
 SSA()
  ODB(SA,3)     CPT(4='+++')
 CAL()    SCB(XX,ODB2)
  IDB(IO,4)
  ODS(EX,2,3,4,5,6,7)
 CMD(BCH,3)
EOQ
```

PDPL Zeitbedarf: für angenommen 60 Entlassungen im Mittel zum Entlassungsdatum und zwar für fünf Tage. Dabei sollen 5 Treffer pro Tag eintreten. Aufsetzen Datenbank Census in 108ms, sequentielles Lesen von fünf Wurzelsegmenten in 104,5ms, sequentielles Lesen von 300 Segmenten des Typs Ceentl in 2820ms und Direktzugriffe in Patsy für 25 Treffer in 1167,5ms. Der Gesamtzeitbedarf ist 4221ms.

SURE:

Suche in der Relation Klinik
BEV +Datum 8_E:
BNL: BNL:
BLV +Datum: BNL: BNL: BNL: BNL:
BID Station_E:
BID +Datum
BKR,VTE: BSR:

Zeitbedarf: 1391ms.
Einlesen der 25 Treffersätze in 620ms.
Bei 13 Suchmoduln sind in zwei Suchläufen aus der Relation IDENT für die Treffersätze die Namen zu ermitteln. Suchumdrehungen kosten 3156ms. Die Direktzugriffe in Ident benötigen 655ms.
Die Gesamtzeit für den Suchrechner beträgt 5822ms.

GPC/DBC: 0,725

3.2.4.2.7 Akkumulierter Laborbericht

Aufgabe:

Sämtliche Labordaten eines Patienten sind auf einer akkumulierten Liste (101,65) darzustellen. Die Druckaufbereitung mit den Umsortierungen soll hier keine Rolle spielen, sondern betrachtet wird das Lesen der Datenbestände. Dabei soll auch die Formatumwandlung für das Analyseergebnis nicht betrachtet werden.

```
SELECT  Izahl, Name, Station, Analysebezeichnung, Ergebnis,
        Faktor, Datum, Pathol, Uhrzeit, Absendestation
 WHERE  Izahl='Izahl'
```

PDPL:

Der hierarchische Pfad des Segmenttyps Apwert liefert alle Informationen. Bei 150 Laborwerten ergibt sich ein Direktzugriff mit 70,2ms und 150 sequentielle Zugriffe mit 1689ms Zugriffszeit. Die Summe ist damit 1759ms.

SURE:

Durchsuchen der Relation Labor mit Bedingung für Izahl und Lesen der Treffersätze in 5370 + 150x32,7 = 10275ms.

GPC/DBC: 0,17

Eine Variante ist das Drucken der akkumulierten Laborlisten für Patienten, die entlassen worden sind von bestimmten Stationen, d.h. die Verbindung dieser Aufgabe mit der vorigen Aufgabe. Diese Listen werden für die Arztbriefschreibung angefordert.

PDPL:

Der Zugriff in Patsy entfällt, da der Name im Apat gespeichert ist. Entlassungslistenpatienten in 3053ms und Labordatenlisten für diese 25 Patienten in 43975ms,
gesamte Zeit 47028ms.

SURE:

Entlassungslisten in 5822ms und Laborlisten für 25 Patienten in 2x5370ms + 25x4905ms.
Gesamte Zeit 133365ms.

GPC/DBC: 0,35

3.2.4.2.8 AD-HOC Anfragen

Unter diesem Anfragetyp sollen alle nicht vom MSH durch spezielle Programme unterstützten Datenverarbeitungsanforderungen verstanden werden. In den meisten Fällen handelt es sich um Aufgaben, bei denen im Datenbanksystem des GPC der gesamte Datenbestand zu durchsuchen ist, weil die Auswahl der Daten nicht über Schlüsselattribute erfolgt.

3.2.4.2.8.1 Einzugsgebiet der Patienten und Aufnahmezeitraum

Aufgabe:

```
SELECT  Izahl, Postlz, Aufn_Datum_Beh
 WHERE  Aufn_Datum_Beh .VOR. 'Datum2' .UND.
        Aufn_Datum_Beh .NACH. 'Datum1'
```

PDPL:

Sequentielle Suche aller hierarchischen Pfade für den Segmenttyp Cobeha. Die Abfragen vor und nach sind in PDPL direkt definiert.
Zeitbedarf 172031x11,0ms = 1892341ms.

SURE:

Das Einzugsgebiet ist in der Relation KLINIK enthalten. Der Datumsvergleich ist nicht definiert. Da die Anfrage im MSH sich auf ganze Jahrgänge bezog, soll angenommen werden, daß das Aufnahmejahr im Suchrechner verglichen wird. Es wird in der Relation jeweils pro Behandlung ein Tupel anzusprechen sein. Da an das Aufnahmedatum bereits zwei Bedingungen gestellt sind, wird über das Entlassungstupel abgefragt. Dadurch wird ein Ergebnisvektor eingespart, noch nicht entlassene Patientenaufenthalte werden erfaßt. Auf jeden Fall sind mehrere Treffervektoren zu erstellen und über geeignete Operationen zum Ergebnisvektor zu überführen.

```
BEV  +Datum 8_E:
BNL:
BGD bzw. BKD für Abfrage Jahresgrenzen:
BNL:
BLV  +Datum:
BNL:  BNL:  BNL:  BNL:
BID  +Datum:
BID  hier Einzugsgebiet angenommen:
BKR,VTE:    BSR:
```

Die Anzahl der zu erstellenden Treffervektoren ist: Anzahl der Bedingungen für das Jahr des Zeitraumes plus Anzahl der Bedingungen für das Einzugsgebiet. Es wird angenommen, daß mit 13 Suchmoduln ein Suchlauf ausreicht. Suchzeit 1391ms.

Die den Ergebnisvektoren zugeordneten Datensätze müssen nicht eingelesen werden, weil der Durchschnitt aller Vektoren, wenn deren Adressen als Mengen aufgefaßt werden, den endgültigen Zielvektor bildet. Falls diese Operation wegen kleinerer Treffermengen und wenig Vektoren im Hauptspeicher erfolgt, ist die Zeit dafür vernachlässigbar. Die Lesezeit der Treffersätze ist AZTx24,8ms.
Die Gesamtzeit 1391ms + AZTx24,8ms.
Die Trefferhäufigkeit THF sei AZT/CARD(KLINIK).

GPC/DBC: 0,627/(THF + 0,000461)

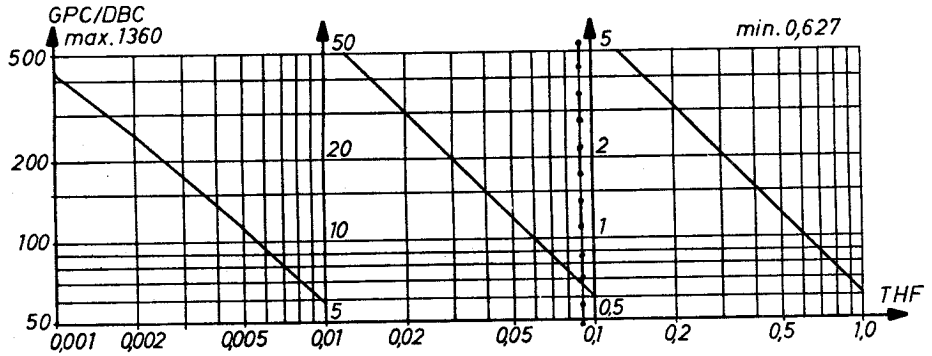


BILD 3.15
Antwortzeitverhältnis GPC/DBC als Funktion der Trefferhäufigkeit.

Wenn der Suchrechner seine Relation sequentiell abarbeitet, dann ist dies verglichen mit dem Einsatz der Suchelektronik ab einer Trefferhäufigkeit von 0,09 günstiger (280764 < 1391 +AZTx24,8). Dieser Wert ist in der Abbildung eingetragen.

3.2.4.2.8.2 Schulpflichtige Ausländerkinder auf Station

Aufgabe:

```
SELECT  Izahl, Namen Station
 WHERE  Staat#'BRD' .UND. Geburts_Jahr   > 1965
        .UND. Entl_Datum_Beh=0
```

Hier wurde eine Vereinfachung bezüglich des Geburtsjahres vorgenommen, da in der Izahl z.B. das Jahrhundert nicht enthalten ist und dafür eine zusätzliche Abfrage in Patsy bzw. IDENT erforderlich gewesen wäre.

PDPL:

Durchlauf durch alle Segmente Leroot mit 935277ms und im Falle der Übereinstimmung von Geburtsjahr und Staat sequentieller Zugriff auf das Segment Lekli für die Prüfung auf bestehenden Aufenthalt mit dem Zeitaufwand von AZTx11,2ms.

SURE:

Der Suchlauf KLINIK mit Bedingung für Geburtsjahr, Aufn_Datum_Stat=Aufn_Datum_Beh und Entlassungsdatum aus der Behandlung=0, d.h. noch nicht entlassen, kostet 1391ms. Der Suchlauf in IDENT mit der Bedingung für die Staatsangehörigkeit und das Geburtsjahr dauert 1579ms. Die Anzahl der Treffer in KLINIK sei AZTkl und die Anzahl der Treffer in IDENT sei AZTid. Es soll angenommen werden, daß AZTid < AZTkl ist. Gelesen werden zuerst die Sätze der Relation mit den wenigsten Treffern. Im Beispiel dauert das Lesen der Treffersätze AZTidx26,2ms. Jetzt werden je 13 Izahlen aus den gelesenen Sätzen in Suchmoduln für die Erstellung der Trefferliste für die Relation KLINIK verwendet. Der Zeitbedarf ist (AZTid/13)x1391ms und führt zu AZT2kl Treffern. Der Durchschnitt der beiden vorliegenden Treffervektoren für die Relation KLINIK ist der endgültige Ergebnisvektor, dessen Datensätze in maximal AZT2klx24,8ms gelesen werden.
Die Gesamtzeit ist <= 1391 + 1578 + AZTid·(26,2+1391/13) + AZT2kl·24,8.

GPC/DBC:

Die Zahl der Lekli Segmente der Datenbanksuche mit der PDPL soll etwa gleich der Anzahl AZT2kl sein. Damit ergibt sich das Verhältnis zu:
(935277+11,2xAZT2kl) / Gesamtzeit für SURE.
Mit der Voraussetzung AZTid/AZT2kl <= 1 und den Annahmen AZTid/AZT2kl=0,5 und AZT2kl=CARD(LEROOT)xTHF läßt sich der Quotient GPC/DBC umformen:

GPC/DBC=(THF + 1,326) / (0,0042 + 8,16·THF)

Eine Auswertung des Quotienten führt auf die Darstellung nach Bild 3.16.

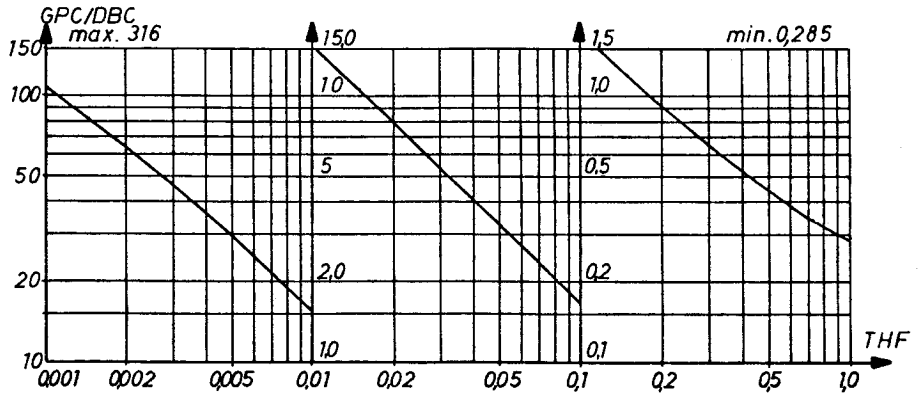


BILD 3.16

Antwortzeitverhältnis GPC/DBC als Funktion der Trefferhäufigkeit.

3.2.4.2.8.3 Fehlerliste, Inkonsistenz redundanter Bitfelder

Aufgabe:

In den Datenbanken werden in den Wurzelsegmenten Bit-Vektoren redundant gespeichert.

```
SELECT  Izahl, Bit_Kette
FROM Coroot
 WHERE  Izahl=.UND. Bit_Kette ≠
   SELECT  Izahl,Bit_Kette
   FROM Leroot
```

PDPL:

Sequentielles Lesen der vom Umfang geringeren Datenbank und Direktzugriffe in der anderen Datenbank. Der Zeitaufwand sequentiell durch Leroot ist 935277. Das direkte Lesen von 65315 Coroot Segmenten kostet 7759422ms, damit ist die Gesamtzeit 8694699ms (ca. 2,5 Stunden Zugriffszeiten).

SURE:

Im Prinzip werden keine redundanten Informationen gespeichert. Eine vergleichbare Aufgabe müßte KLINIK sequentiell in 280764ms lesen und je 13 Patienten über SURE in IDENT prüfen.
Die Zeit für 62990 Vergleiche in IDENT ist (62990/13)x1578ms.
Die Treffervektoren enthalten Adressen von Sätzen, die nicht übereinstimmen. Das Einlesen der Treffer kostet AZIx26,2 ms.
Die Gesamtzeit ist(7926781 + AZTx26,2)ms.

GPC/DBC: 1,09 für AZT=0

3.2.4.2.8.4 Diagnose: Klinik,Org_Nr, Diag_Datum, Doku_Datum, Arzt

Aufgabe:

```
SELECT Izahl, Diagnose, Klinik, Org_Nr, Diag_Datum,
       Doku_Datum
 WHERE Diagnose=Diagnosecode
```

PDPL:

Durchlauf durch die Daten des hierarchischen Pfades für den Segmenttyp Diag. Die Invertierung, die im MSH verfügbar ist, wird von GURS zur Zeit nicht angesprochen. Deshalb ist die sequentielle Zeit mit 3360000ms unvergleichlich höher, als sie sein müßte.

SURE:

Die Treffervektorerstellung dauert 3875ms. Bei AZT Treffern ist die Gesamtzeit 3875+AZTx31,9.

GPC/DBC:

Die Aufgabe führt wie in 3.2.4.2.5.1 auf das Verhältnis 0,419 / (THF + 0,000483). Die Auswertung entspricht damit Bild 3.13.
Sequentielles Lesen mit dem Suchrechner ist ab etwa THF < 0,07 günstiger als die Ermittlung der Treffer mittels Suchlauf und Direktzugriffen (560915 <= 3875 + AZT x 31,9).

3.2.4.2.8.5 Wahlleistung 'Arzt' auf Stationen

Aufgabe:

```
SELECT  Izahl, Aufn_Nr, Aufn_Dauer, Tarif_Kl, Station,
        Aufn_Jahr
 WHERE  SUBSTR(Tarif_Kl,1,1)='S' .UND. Station=xyz .UND.
        Aufn_Jahr= 1978
```

PDPL:

Sequentielles Lesen aller Segmente Leaufn in 10,8msx87165= 941382ms.
Falls die Bedingung Aufn_Jahr=78 erfüllt ist, werden alle abhängigen Lekli Segmente gelesen und nach Tarifklasse und Station überprüft. Die Anzahl der Segmente für 1978 ist angenommen 0,25 aller Lekli Segmente, so daß die Lesezeit 11,2msx(121737x0,25)=340864ms beträgt. Unabhängig von der Anzahl der Treffer sind also 1282246ms Zeit aufzuwenden.

SURE:

Der Suchlauf in der Relation KLINIK benötigt 1391ms. Das Lesen der Treffersätze dauert AZTx24,8ms.

GPC/DBC:

Mit der Trefferhäufigkeit in Klinikaufenthalten von THF=AZT/121737 wird GPC/DBC=0,425/(THF+0,00046)
Die Auswertung führt auf die Darstellung in Bild 3.17.

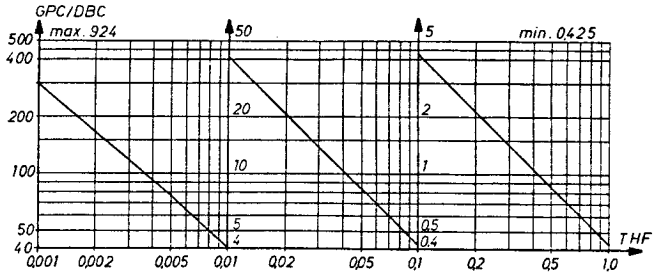


BILD 3.17

Antwortzeitverhältnis GPC/DBC als Funktion der Trefferhäufigkeit.

Der Sequentielle Modus des Suchrechners ist gegenüber dem Suchelektronik Modus ab einer THF >= 0,09 schneller.

3.2.4.2.8.6 Folgestation bei Voraufenthalt größer 3 Tagen

Aufgabe:

Es gibt eine sog. Aufnahmestation, auf der Patienten im allgemeinen nur einen Aufenthaltstag haben sollten. Gesucht sind in 1978 die Folgestationen, bei denen Patienten unmittelbar vorher länger als 3 Tage auf der Aufnahmestation gewesen sind.
Das Ergebnis ist durch die Abhängigkeit zweier Tupel derselben Relation voneinander bestimmt, wobei das Entlassungsdatum der einen das Aufnahmedatum der anderen bedeutet. Damit ist die Anfrage in Sequel in zwei Schritten über eine Zwischenrelation zu lösen:

1.
```
SELECT  Izahl, Station, Aufn_Dauer, Entl_Dat_Stat,
        Aufn_Jahr
FROM KLINIK
INTO Temprel (Izahl, Temp_Stat,Temp_Dauer, Auf_Dat_Station
              Aufn_Jahr)
 WHERE  Station=xyz .UND. Aufn_Dauer   <  3 .UND. Aufn_Jahr=78
```

Es soll angenommen werden, daß die Funktion INTO eine Ergebnisrelation erstellt, die die Attributwerte in der Reihenfolge des übergeordneten SELECT übernimmt, jedoch als ansprechbare Attributnamen die in der Klammer angegebenen Namen vorsieht.

2.
```
SELECT  Station, Izahl, Auf_Dat_Station
FROM KLINIK
 WHERE  Auf_Dat_Station=, Izahl=
    SELECT  Auf_Dat_Station, Izahl
    FROM Temprel
```

PDPL:

Hier ist die Ordnung der Segmente Lekli gleichzeitig eine zeitliche Reihung, in der der jüngste Klinikaufenthalt zuerst gefunden wird. In IMS kann das letzte Segment einer Folge angesprochen werden, da die PDPL dies nicht vorsieht, müssen alle Kliniksegmente gelesen und jeweils mit dem vorausgegangenen verglichen werden.

```
DDC(Leipat,Izahl,Izahl,Aufn_Datum_Beh, Aufn_Datum_Beh ,
   Station, Station, Aufn_Dauer, Aufn_Datum_Station,
    Entl_Datum_Station)
SFS(1,Dummy)
 SSA()
  CPT(1=Initialwert) CPT(3=Init) CPT(7='02')
  ODB(IO,1,3)
 CMD(BCH,2)
 CAL()
```

```
SFS(2,Lekli)
 SSA()
 CAL()   SCB(XX,STOP)
  IDB(IO,2,4,6,8,9)
  CPT(8<>9) CPT(7=8)
  WHR(6='    1'.UND.7  > '    3'.UND.9       +' O')
   CMD(LGC,O,BCH,3)
  ODB(EX,1,2,3,5,7)
SFS(3,Dummy)
 SSA()
  CPT(1=2) CPT(3=4)   CPT(5=6)
 CMD(BCH,2)
 CAL()
EOQ
```

Die Zeit ist 11,7 ms · 121737 = 1424323ms.

SURE:

Die Aufenthaltsdauer ist entweder als Attribut verfügbar oder muß durch ein Programm im Suchrechner ermittelt werden. Im ersten Fall werden zur Erstellung der Zwischenrelation Temprel 1391+AZTx24,8+AZTx20ms benötigt. Dabei wurden für das Schreiben der Temprel mit DAM 20ms je Treffer angenommen. Im zweiten Fall erfordert das sequentielle Lesen 280764ms statt 1391+AZTx24,8. Diese sequentielle Zeit ist ab einer Trefferzahl von 6200 (THF=0,05) schneller.
Es sind AZT/13 Treffervektoren zu erstellen, deren Datensätze in einer Zeit von AZTx24,8ms eingelesen werden. Die Treffervektorerstellung dauert mit dem sequentiellen Lesen der Temprel etwa (AZT/13)x1391 + 2xAZT.

GPC/DBC:

1424323/1391+AZT(24,8+20)+(AZT/13)x1391+AZTx(2+24,8))
Mit der Annahme THF=AZT/CARD(KLINIK) vereinfacht sich der Quotient zu:
0,0655/(0,0000639+THF)
Die Auswertung dieses Ausdruckes führt zur Darstellung nach Bild 3.18.

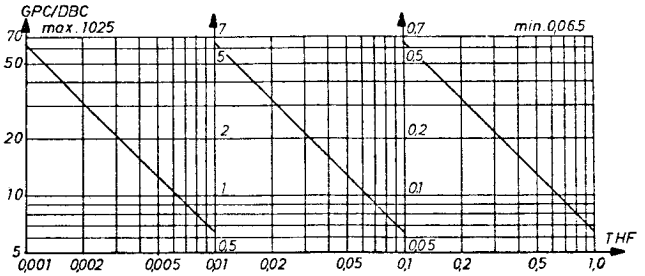


BILD 3.18

GPC/DBC=f(Trefferhäufigkeit)

3.2.4.2.8.7 Diagnose und Laborwerte

Aufgabe:

```
SELECT  Izahl, Diagnose, Laboranalyse, Laborergebnis
 WHERE (Diagnose=d1 .ODER. Diagnose=d2 .ODER. Diagnose=d3)
        .UND. DIAGNOSE≠d4
        .UND. (Laboranalyse=L1 .ODER. ... .ODER. Laboranalyse=L10)
```

Bemerkung:

Die Diagnosen können bezogen sein auf Patienten mit all ihren Behandlungen oder auf Behandlungen mit all ihren Patienten. Im zweiten Fall werden bestimmte Patienten mehrfach gewertet. Hier soll auf Patienten bezogen werden.

PDPL:

Die Invertierung für Diagnosen ist nicht als Schnittstelle in GURS verfügbar. Entweder werden Laborwerte oder Diagnosen sequentiell verarbeitet und bei Erfüllheit der Bedingung über Direktzugriff in der jeweils anderen Datenbank die jeweils anderen Bedingungen geprüft. Das Lesen der Laborwerte kostet 2,7 mal so viel Zeit wie das Lesen der Diagnosen.

1. Lesen aller Diagnosen in 3360000ms.
   Die Trefferrate bei j=1600 und CARD(Diag)=251497 ist je Elementarbedingung 0,0006, so daß (3x0,0006)x0,9994 die Gesamttrefferrate und die Trefferanzahl AZT=453 ist.
   (Diese Annahmen entsprechen denen aus 3.2.4.2.2).

2. Lesen der im Mittel 30 Laborwerte durch Direktzugriff für das Wurzelsegment Aproot in 453x70,2 Fällen in 51801ms und sequentielle Lesen der angenommenen Zahl von 453x30 Laborwertssegmenten in 453x30x11,26ms= 153023ms.
   Die ermittelte Anzahl von Treffern 453 soll gleich der Anzahl der Patienten sein, was nicht sicher ist, weil die Diagnosen nicht gleichverteilt sind, die Diagnosehäufigkeiten oft nicht voneinander unabhängig sind und falls Diagnosen gespeichert sind, pro Patient im Mittel 3,5 Diagnosen pro Behandlung gestellt wurden.

3. Die Gesamtzeit beträgt: 3544824ms.

SURE:

1. Das Erstellen der Treffervektoren für die drei ersten Diagnosebedingungen erfolgt in 3875 ms.

2. Lesen der Menge M1=M(d1)+M(d2)+M(d3) in 14451ms. Die Anzahl der Treffer wurde mit 3xTHFxCARD(Diag)=453 abgeschätzt.

3. Die Menge M(nicht d4) wird nicht eingelesen, sondern in 453/13 Suchläufen werden diejenigen Patienten von M1 ermittelt, die mindestens eine Diagnose d4 besitzen und damit aus M1 zu streichen sind. Diese Suchzeit beträgt 35x3875ms= 135625ms, das Entfernen der Treffer erfolgt im Hauptspeicher. Die Anzahl der zu entfernenden Treffer ist 453x0,0006=0,27, also maximal einer.

4. Lesen der Relation Labor für die Treffervektorerstellung 452/13 mal je Laboranalyse erfordert 35x5370ms=1879500ms. Es soll wie bei der PDPL angenommen werden, daß 30 Laborwerte je Patient gefunden werden und einzulesen sind in 443412ms. Der Mengenverbund über die Izahlen soll zeitlich nicht gewertet werden.

5. Die Gesamtzeit beträgt 2476863ms.

GPC/DBC: 1,43

Zusatz:

Dieses Beispiel soll dazu benutzt werden, um den GPC/DBC-Faktor bei konstanter Zahl von Laboranalysen, aber variierender Zahl disjunkt verbundener Diagnosebedingungen (Komplexität der Anfrage) zu ermitteln (ohne die Ausschlußdiagnose d4).

PDPL: wie vorher

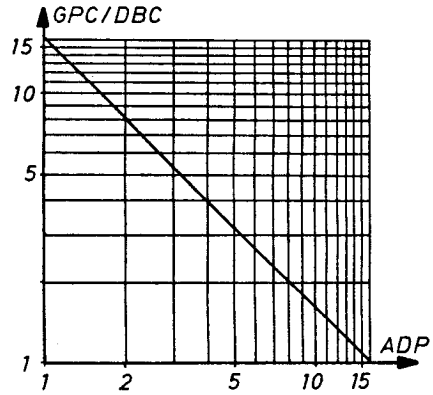
Sure:

Die Anzahl der Diagnosebedingungen sei ADP. Die Treffervektorzeit in der Relation DIAG ist (ADP/13)x3875ms, wobei ADP/13 auf ganze Zahlen aufzurunden ist. Die Lesezeit beträgt bei AZT=251497x0,0006xADP Treffern AZTx31,9ms.
Die Treffervektorerstellung in der Laborrelation benötigt (AZT/13)x5370ms, wobei AZT/13 wieder auf ganze Zahlen aufzurunden ist. Es soll angenommen werden, daß 30 Laborwerte im Mittel je AZT gefunden werden, so daß AZTx30x32,7ms Lesezeit benötigt werden. Der Mengenverbund über die Izahl soll zeitlich nicht gewertet werden.

Gesamtzeit: ADPx(3875/13)+AZTx(31,9+5370/13x30x32,7)ms.

GPC/DBC:

Durch entsprechende Umformung und Ausrechnung ergibt sich 16,07/ADP.



Darstellung des GPC/DBC-Faktors in Abhängigkeit von der Anzahl disjunkt verbundener Diagnosebedingungen einer sonst konstanten Datenbankanfrage.

BILD 3.19

3.2.4.2.8.8 Anzahl gleicher Diagnosen >1 in derselben Dokumentationseinheit.

Aufgabe:

```
SELECT  Izahl, Diagnose, Aufn_Datum_Beh, Dokum_Datum
 FROM Diag
 INTO Temprel(Izahl, Diag, Aufn_Datum, Dokum_Datum)
```

und als zweiten Teil:

```
SELECT  Izahl, Diagnose, Aufn_Datum_Beh, Dokum_Datum
 FROM Temprel
 WHERE  Izahl=, Diagnose=, Aufn_Datum=, Dokum_Datum=
  SELECT  Izahl, Diagnose, Aufn_Datum, Dokum_Datum
  FROM Diag
  WHERE  COUNT(Dokum_Datum    > 1)
```

COUNT soll die Anzahl der gefundenen Tupel aus Diag je Tupel aus Temprel ermitteln.

PDPL:

Die Darstellung ist von den Lesesequenzen her kompliziert und wäre sinnvollerweise mit einem PL/1-Programm wohl mit weniger Aufwand für die Datensuche zu erstellen. Zur Erläuterung dient Bild 3.20. Es werden die Segmenttypen COROOT, COBEHA und COQUAL je Patient (Pfade 1 bis 3) gelesen und automatisch die Schlüssel im Schlüsselbereich gerettet (in GURS). Jetzt wird (Pfad 4) das nächste CODIKO Segment gelesen und die Felder im Datenelementspeicher für den Vergleich gespeichert. Der Vergleich erfolgt mit allen weiteren Segmenten des Pfades 5. Sobald zu der Dokumentationseinheit (Segmenttyp Coqual) keine weiteren Segmente gefunden werden (Statuscode=GE), verzweigt die Ausführung zu Pfad 6, der das verglichene Segment direkt wieder einliest. Danach wird das nächste Segment in Pfad 4 eingelesen und wieder mit dem jetzt verbleibenden Rest verglichen. Bei Treffer erfolgt eine Ausgabe.

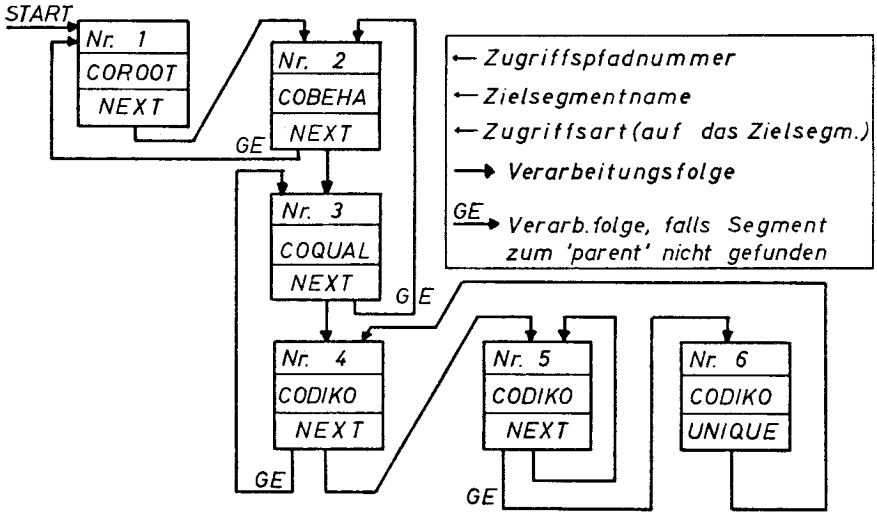


BILD 3.20 *Darstellung der Zugriffspfadsequenz; nähere Erläuterungen befinden sich im vorangehenden und nachfolgenden Text.*

Die Darstellung der Zugriffspfadsequenz für die beschriebene Aufgabe erfolgt in Form von Rechtecksymbolen je zu definierenden PDPL-Pfad, die die Pfadnummer, den Segmenttyp und die Zugriffsfunktion enthalten. Normale Verzweigungen sind nicht beschriftet, Verzweigungen auf Grund Zugriffsstatuscode GE (Datensatz nicht gefunden) sind durch diesen gekennzeichnet.

Der Zeitaufwand ist: Wurzelsegmente 1065883ms, Cobeha-Segmente 1723751ms, Coqual-Segmente 961696ms und bei im Mittel 4 Codiko je Coqual-Segmenttyp ca. 13216433ms für die Codiko-Segmente.
Die Summe beträgt 16967762ms.

Sure:

Sequentielles Lesen der Relation DIAG: 560915ms.
Es folgt je 13 Sätzen die Prüfung gegen die Relation DIAG, wobei auf Identität Izahl, Behandlung, Dokumentationseinheit und Diagnose geprüft wird. Bei mehr als einem Treffer liegt ein gesuchter Fall vor. 251497/13 Suchläufe erfolgen in 74965451ms.
Die Gesamtzeit ist 75511122ms.

GPC/DBC: 0,225

3.2.4.2.8.9 Nur eine von 6 Diagnosen

Aufgabe:

Von 6 Diagnosen d_1 bis d_6 soll ein Patient, falls er eine davon aufweist, keine der restlichen 5 Diagnosen besitzen (5 Ausschlußdiagnosen).

Diagnosen des Patienten : $D = \{D_1, D_2, D_3, \ldots, D_m\}$

Eine beliebige Diagnose des Patienten : $D_i \in D$

Die restlichen Diagnosen des Patienten: $D^* = D - D_i$

Gegebene 6 Diagnosen : $d = \{d_1, d_2, d_3, d_4, d_5, d_6\}$

Trefferbedingung : $D_i \in d$ und $d \cap D^* = \emptyset$

PDPL:

Das Lesen aller Diagnosesegmente ist mit 3360000ms erforderlich. Die Auswerte logik für das laufende Zählen von Treffern in 6 Trefferzählervariablen wird in drei Extrapfaden durchgeführt. Da dieselbe Diagnose bei einem Patienten mehrfach auftreten kann, reicht es nicht aus, die Anzahl Treffer > 1 als Anzeichen für Nichterfülltheit der Anfrage zu werten. Deshalb wird es nötig, weitere drei Pfade zu benutzen, um bei Izahlwechsel abzuzählen, ob mehr als einer oder keiner der 6 Tefferzähler ein Ergebnis > 0 aufweist.
Dieser Aufwand ist immer noch gering gegenüber einer PL/1-Lösung.

SURE:

Die Treffervektoren für alle 6 Diagnosen werden in 3875ms gewonnen und die etwa 0,0006x6x251497=906 Treffer in 28901 eingelesen. M_i, M_j sind die Izahlmengen, zugehörig zu den 6 Diagnosen d_1 bis d_6 mit $1 <= i <= 6$ und $1 <= j <= 6$. Die sechs Relationen zu je 151 Tupeln werden nach dem folgenden Algorithmus bereinigt:

$$\boxed{\overline{M_i} \cap (\bigcup_{j=1}^{6} M_j)} \cap \overline{M_i} \quad \text{unter Auslassung } M_j \text{ für } i=j$$

Diese Operation kann im Hauptspeicher erfolgen, so daß die Zeit bei dieser Izahlmenge noch vernachlässigt ist.
Gesamtzeit: 32776ms.

GPC/DBC: 102,5

3.2.4.2.8.10 Dieselbe Diagnose in Folgebehandlungen

Aufgabe:

```
SELECT  Izahl, Diagnose
 WHERE  Diagnose=d1
        .UND.
        Anzahl pro Patient    > 1
```

PDPL:

Die Codiko-Segmente werden sequentiell durchlaufen und Treffer gezählt. Bei jedem Izahlwechsel wird geprüft, ob der Zähler > 1 ist. Es erfolgt Initialisierung und gegebenenfalls die Trefferausgabe.
Der Zeitbedarf ist 3360000ms.

Sure:

Die Treffervektorerstellung für die Diagnose d1 erfolgt in 3875ms. Das Einlesen der ca. 251497x0,0006=151 Sätze erfordert 4817ms.
Die Gesamtzahl ist 8692ms.
Im Hauptspeicher ist zu prüfen, ob Izahlen mehrfach auftreten. Da die Trefferzahl der Diagnose stark streut, ist der GPC/DBC-Faktor in Bild 3.21 für angenommene Treffer angegeben. Das Verhältnis GPC/DBC führt auf die Funktion 105329/(121+AZT).

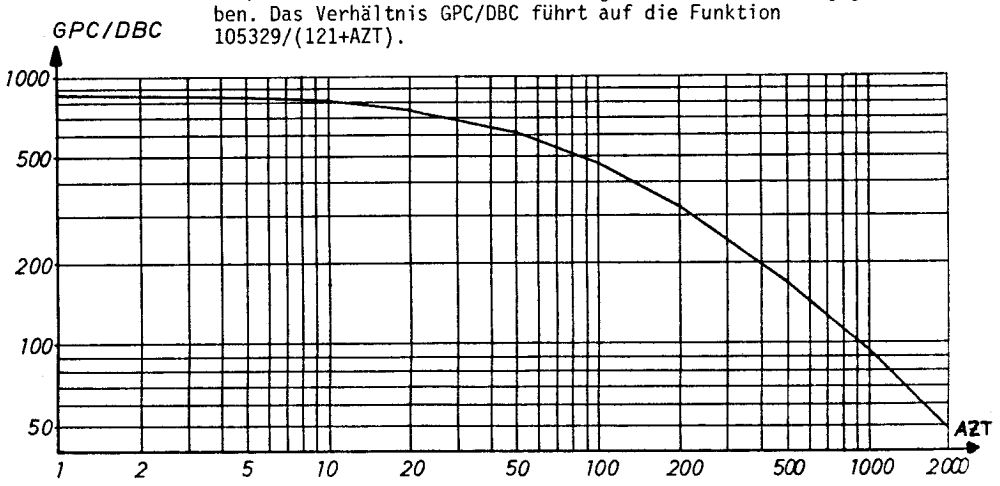


BILD 3.21

Antwortzeitverhältnis GPC/DBC als Funktion der Anzahl der Diagnosetreffer bei Patientensuche über die ganze Datenbank.

4. Zusammenfassende Diskussion

Die Arbeit beschreibt eine im Medizinischen System Hannover (MSH) entwickelte Datenbankverarbeitungssprache (PDPL) und vergleicht die Leistungsfähigkeit des verwendeten Mehrzweckrechners (GPC = general purpose computer) mit der des Suchrechners SURE der Universität Braunschweig (DBC = data base computer). Der Vergleich wurde auf der Basis von Datenbeständen und Arbeitslasten des MSH durchgeführt.

Die Genauigkeit der Ausrechnungen wurde oft mit drei Stellen nach dem Komma angegeben. Diese Genauigkeit entspricht nicht der Grundgenauigkeit der eingehenden Faktoren. Insbesondere Betriebssystemanteile sind für das verwendete Datenbankverwaltungssystem IMS von vielen Einflußgrößen abhängig, die in die durchgeführten Ausrechnungen nicht eingehen. Alleine die Pufferverwaltung mit dem Problem der 'buffer pool compaction' (28) führt auf nicht generalisierbare Antwortzeitabhängigkeiten auf Grund der Daten und der Zugriffsfolge selbst, so daß die in dieser Arbeit ermittelten Werte nur von ihrer Größenordnung her verallgemeinerbar sind. Auch beinhalten die Arbeitslasten nicht besonders komplizierte und vom Umfang her zeitkritische Mengenoperationen. Die Mengenoperationen werden als im Hauptspeicher vernachlässigbar schnell durchführbar eingestuft.

Der Vergleich des Speicheraufwandes bei Vernachlässigung der Invertierungen im Mehrzweckrechner ist etwa GPC/DBC=1,2. Bei Einbeziehung der Invertierungen, deren Zahl nicht generalisierbar ist, steigt der Aufwand in MSH schätzungsweise bis auf GPC/DBC=2.

Die Arbeitslasten enthalten folgende Sucharten:

1. Suchattribut mit Schlüsseleigenschaft
2. Suchattribut ohne Schlüsseleigenschaft
3. gemischte Suche.

Ein Problem für den Suchrechner war die Suche nach abgeleiteten Größen, weil dies ohne Datenmanipulation in den Suchmoduln grundsätzlich zum sequentiellen Lesen des gesamten Bestandes und Verarbeiten mit einem extra Programm führt.

Ein weiterer nachteiliger Gegensatz zur GPC-Lösung ist das Nicht-Vorliegen von physischen Ordnungsbeziehungen der Datensätze, so daß fertige Ergebnislisten z.B. durch Umsortierungen in die endgültige Reihung gebracht werden müssen.

Problematisch ist, daß der Verwaltungsüberhang, den auch der Suchrechner als Datenbankrechner haben wird, nicht in den Betrachtungen enthalten ist, während der entsprechende Anteil beim Mehrzweckrechner enthalten ist. Es kann aber davon ausgegangen werden, daß der Aufwand gegenüber IMS um mehr als eine Größenordnung darunter liegt, so daß das Gesamtergebnis dieser Untersuchung grundsätzlich nicht in Frage zu stellen ist.

Insgesamt bringt die Einführung der PDPL besonders für die Formulierung und Durchführung von Ad hoc Anfragen Vorteile im MSH. Im Vergleich der Leistungsfähigkeit schneidet die PDPL-Lösung mit IMS Datenbanken besser ab, als die Implementierung eines relationalen Systems im Mehrzweckrechner. Die Aussage (6), daß der Datenbankrechner grundsätzlich bei einer Trefferzahl größer 50 für den Gleichheitsoperator viel besser als ('much better than') der Mehrzweckrechner ist, gilt für den Vergleich PDPL mit SURE nicht. Rein qualitativ hat sich die Vermutung bestätigt, daß die speziellen Zugriffswege den Mehrzweckrechner schneller machen als den Datenbankrechner und daß bei fehlenden schlüsselgestützten Zugriffen und geringer Trefferrate der Suchrechner dem GPC weit überlegen ist.

Das Ziel der Arbeit war auch die Quantifizierung dieser Aussagen im Vergleich für ausgewählte Arbeitslasten. Eine allgemeine Lösung des Problems Zugriffszeitermittlung war nicht möglich. Dennoch können einige Verallgemeinerungen auf Grund der Ergebnisse gemacht werden:

Durch Umordnung der Beispiele nach dem GPC/DBC-Verhältnis in Tabelle 3.15 lassen sich drei Gruppen erkennen:

Gruppe 1:

Von den Suchkriterien her liegt der GPC/DBC-Faktor bei Verwendung von Schlüsselattributen durch den Mehrzweckrechner, ohne zusätzliche Zugriffsketten, im Mittel bei 0,06. Der Mehrzweckrechner ist nur in der Größenordnung 35 bis 15 mal so schnell, trotz spezieller Zugriffswege. Bei Langläufern oder interaktiven Programmen kann dieser Faktor allerdings schon eine entscheidende Größenordnung sein.

Gruppe 2:

Sobald Kettenzugriffe, z.B. GET NEXT mit folgendem GET UNIQUE, auf Seiten des GPC erforderlich werden, ändert sich das GPC/DBC-Verhältnis trotz Nutzung von Schlüsselwegen um den Faktor 10 zu Ungunsten des Mehrzweckrechners:
Die beiden Konzepte sind etwa gleich schnell, bei leichtem Vorteil für den GPC.

Gruppe 3:

Sind die Suchkriterien keine Schlüsselattribute, so ist die Trefferrate entscheidend. Bei Trefferraten je nach Arbeitslast in der Größenordnung von 40% ist der GPC leicht im Vorteil (bis doppelt so schnell), da der Suchrechner über Suchläufe und zugehörige Direktzugriffe arbeiten muß. Sind dagegen wenige Treffer vorhanden, so ergibt sich, daß der Datenbankrechner in der Größenordnung 900 mal schneller ist.

Es muß festgestellt werden, daß die Trefferrate, bei der der Suchrechner besser nicht über Suchlauf und Direktzugriff, sondern ohne Suchelektronik sequentiell arbeitet, schon in der Größenordnung 0,1 liegt. Ob diese Betriebsart bei der Bearbeitung einer Vielzahl von Anfragen sinnvoll durchgeführt werden kann, ist nicht sicher. Einschließlich der Möglichkeit der sequentiellen Verarbeitung ist der Suchrechner für Nicht-Schlüsselattribute in keinem Fall langsamer als der Mehrzweckrechner.

Anfrage	Suchkriterium GPC		Ausführungsproblem		GPC/DBC
	Schlüssel	nicht-Schlüssel	Ketten-zugriff	Verbund u.a. Mengenoper.	
3.6	+	-	-	-	0,03
4.3	+	-	-	+	0,05
4.6	+	-	-	-	0,05
4.7	+	-	-	-	0,06
3.2	+	-	-	-	0,07
4.2	+	-	-	-	0,07
4.4	+	-	-	-	0,07
4.5	+	+	-	+	0,07
3.5	+	-	+	+	0,2
7A	+	-	-	-	0,2
8.8	-	+	+	+	0,2
3.1	+	+	+	+	0,3
4.1	+	+	+	+	0,3
7B	+	-	+	+	0,3
3.4	+	+	+	+	0,6
6	+	-	-	+	0,7
8.3	+	+	+	+	1,0
8.7A	+	+	+	+	1,4
8.7B	+	+	+	+	1 - 6
8.9	-	+	-	-	100
8.2	-	+	-	+	30 - 300
3.3	-	+	-	-	800
5.2	-	+	-	-	0,2 - 900
5.1	-	+	-	-	0,4 - 900
8.4	-	+	-	-	0,4 - 900
8.5	-	+	-	-	0,4 - 900
8.10	-	+	-	-	1,5 - 900
8.6	-	+	+	+	0,07 - 1100
8.1	-	+	-	+	0,6 - 1400

TABELLE 3.15

Gruppenbildung des GPC/DBC - Antwortzeitverhältnisses

In Sonderfällen, wie z.B. 4.5 oder 8.8 mit erhöhtem Leseaufwand, kann mal die eine, mal die andere Lösung besser sein. Generell ist festzustellen, daß der Suchrechner mit Suchelektronik bis in die Größenordnung 1000 schneller sein kann, während der Mehrzweckrechner nur bis in die Größenordnung 100 schneller sein kann. Diese Aussagen stimmen in der Tendenz auch mit Banerjee (6) überein.

Wenn bestimmte zeitaufwendige Funktionen für die Durchführung von Mengenoperationen oder Sortierungen durch weitere Spezialprozessoren gestützt würden, ist der Datenbankrechner eine echte Alternative zu Datenbanksystemen, da bei fast gleicher oder besserer Leistung der Hauptrechner spürbar entlastet wird. Das System /38 Datenbankrechner der Firma IBM ist von der Konzeption ein erster Schritt im kommerziellen Bereich dorthin, obwohl hier noch keine Spezialelektronik auf der Sekundärspeicherseite eingesetzt wird. Echte Hardwarekonzepte wie die Suchrechnerentwicklung in Braunschweig oder neuere Techniken der Bubblespeicher werden die Richtung zu Spezialrechnern mit relationaler Datenbankverwaltung erfolgreich weitertreiben. Solange solche Rechner und relationale Datenverwaltungssysteme nicht generell verfügbar sind, haben Sprachergänzungen, wie die PDPL, jedoch ihre volle Berechtigung:
Sie bieten den nötigen Komfort, Benutzerwünsche leichter erfüllen zu können, auch wenn die Antworten für spezielle Wünsche 1000 mal langsamer als in Zukunft gegeben werden.

5. LITERATURVERZEICHNIS

(1) K. Alber,
Language interface
und
From problem to program,
Vorträge Medizinische Hochschule Hannover,
Advanced course on Informatics and Medicine, 1975

(2) ANSI/X3/SPARC,
Study group on data base management systems
Interim report, Washington DC: CBEMA 1975

(3) K. Appel, M.Jainz, T. Risch, K. Sauter, W. Schneider,
A data manager for the health information system
Berlin,
Computer Progr. Biomed., 6(1976)

(4) M.M. Astrahan, D.D. Chamberlin,
Implementation of a structured english query language,
Communications of the ACM, Vol 18, No 10, 1975

(5) M.M. Astrahan, et.al.,
System R: Relational approach to data base management,
ACM Transactions on data base systems, Vol 1, No 2,
June 1976

(6) J. Banerjee, D.K. Hsiao,
Performance of a data base machine in supporting
relational data bases,
Proceedings 'Very large data bases', Berlin,
13.-15.Sept. 1978

(7) Bayerische Staatskanzlei München,
Organisation und Test der Grundstücksdatenbank,
IMS/VS (1976 und IMS/VS-KDBS (1978)

(8) Bundes- und Niedersächsisches Landesdatenschutzgesetz

(9) A. Blaser, H. Schmutz,
Datenbankforschung heute - Versuch einer Bestandsaufnahme,
IBM Nachrichten, 26. Jahrg., Heft 231 und 232, 1976

(10) Bundesministerium des Innern,
Kompatible Anwendungsprogrammierung, Schnittstellen-
beschreibungen KDBS, KDCS und KSDS

(11) P. Bogenstätter, et.al.,
Variable Befunderfassung und -verarbeitung mit dem
allgemeinen dialogfähigen System für Datensichtgeräte
CLIST,
Symposium 'Klartextverarbeitung medizinischer Befunde ...'
der Sektion Klartextanalyse der Arbeitsgruppe Medizinische
Informatik der DGMDS und dem medizinischen Rechenzentrum
der Universität Wien, 23.06.1975

(12) Control Data Institut,
Kongreß für Software Management,
'Strukto 78', Frankfurt, 24.-26.10.78

(13) D.D.Chamberlin, et.al.,
SEQUEL 2: A unified approach to data definition,
manipulation and control,
IBM Journal of research and developement, Nov. 1976

(14) CODASYL,
Data description language,
Journal of development, 1973

(15) E.F. Codd,
A data base sublanguage founded on the relational
calculus,
ACM SIGFIDET Workschop on data description, access
and control, San Diego, Cal., Nov. 1971

(16) W. Dreger,
Grundlagen der Systemtechnik,
IBM Nachrichten, Nr. 218 und 219, 1973

(17) R.J. Dubien, H.D. Covvey, M.C. Sevcik, E.D. Wigle,
A data base system implementation providing data inde-
pendence for medical applications,
Proceedings Medinfo 77, Toronto, 1977

(18) W. End, H. Gotthardt, R. Winkelmann,
Softwareentwicklung, Siemens Verlag, ISBN
3-8009.1235-X, Berlin/München, 1976

(19) U.E. Fischer,
Neue Methoden und Techniken der Programmierung,
Überblick und Entwicklungstendenzen,
IBM Nachrichten Nr. 222, 1974

(20) J.W. Forrester,
Grundzüge einer Systemtheorie,
Betriebsw.. Verlag Dr. Th. Gabler, Wiesbaden, 1972

(21) G. Friedrich,
Simulation von DV-Anlagen, Planerische Ermittlung von
Systemkenngrößen,
data report, Heft 1, 1976

(22) C. Frasson,
A System to increase data independence in a hierarchical
structure,
In: G. Goos, J. Hartmanis (edtrs), Lecture notes in
Computer Science, Vol 34, Springer Verlag, Berlin, 1975

(23) S.H. Fuller, F. Baskett,
An analysis of drum storage units,
Journal of the ACM, Vol 22, No 1, Jan. 1973

(24) K. Gewald, G. Haake, W. Pfadler,
Software Engineering, Grundlagen und Technik rationeller Programmentwicklung,
R. Oldenbourg Verlag, München/Wien, 1977

(25) GMD,
Erfahrungsbericht 1975, Performance von IMS

(26) G. Goos,
Basic notions of informatics,
Vortrag, Medizinische Hochschule Hannover, Advanced course on Informatics and Medicine, 1975

(27) G. Gordon,
Systemsimulation,
R. Oldenbourg Verlag, München, 1972

(28) L. Granegard,
Monitoring performance of IMS batch programs,
Guide Munich, June 1974

(29) M. Griffiths,
Some aspects of system programming,
Vortrag, Medizinische Hochschule Hannover, Advanced course on Informatics and Medicine, 1975

(30) Universität Heidelberg,
Studienplan zum Studiengang Medizinische Informatik

(31) M. Heinz,
Einfluß der Hardware-Konfiguration und des Aufgabenprofiles auf den Durchsatz eines Rechnersystems,
Angewandte Informatik, Heft 6, 1976

(32) H.L. Hill,
Data base system evaluation,
Vortragsmanuskripte, Informatik Symposium IBM Deutschland, 1975

(33) E. Hirsch,
Software Engineering - Projektbegleitende Dokumentation mit BYBLOS 2000,
data report, 12. Jahrg., Heft 5, 1977

(34) J. Hofmann, F.J. Leven,
Diplominformatiker der Medizin - ein neues Berufsbild im Gesundheitswesen,
medizinische technik, 96.Jahrg., Heft 6, 1976

(35) J. Hofmann,
Ein neuer Studiengang: Medizinische Informatik,
Angewandte Informatik, Nr. 1, 1976

(36) F. Hossfeld,
A System identification access to computer performance analysis,
Angewandte Informatik, Heft 4, 1978

(37) IBM,
Information Management System /360 (IMS),
Version 2, General Information Manual, IBM Publication GR 20-o/65-1

(38) IBM
System /360 Operating System, Supervisor and Datamanagement Services,
IBM Publication C 28-6648

(39) IBM
OS PL/1 Optimizing Compiler, Programmers Guide,
IBM Form SC33-0006, 1974

(40) IBM
HIPO - Hierarchy plus Input, Process, Output,
Installation Management GC 20-1851-1, 1974

(41) IBM,
OS PL/1 checkout and optimizing compiler: language reference manual,
IBM Form Sc 33-0009-2, 1972

(42) IBM,
Information Management System /360, system application guide,
Version 2, IBM Publication SH 20-0910-4, 1974

(43) IBM,
IQF - interactive query facility for IMS /360,
Version 2, Form Nr. CH 20-1074.1

(44) W. Jentsch,
Digitale Simulation kontinuierlicher Systeme,
R. Oldenbourg Verlag, München, 1969

(45) A.U. Jones,
A user oriented data base retrieval system, DBRS,
IBM Systems Journal, No 1, 1977

(46) I. Karlowsky, H.-O. Leilich, G. Stiege,
Ein Suchrechner für Datenbankanwendungen,
Elektronische Rechenanlagen, 17.Jahrg., Heft 3, 1975

(47) V. Kevin, M. Whitney,
Relational data management: poison or panacea.,
General Motors Research Publication, GMR-1553, Warren, Michigan, 1974

(48) J. Klonk, K. Sauter, W. Weingarten,
Anwendungsorientiertes Design einer nicht-prozeduralen Retrievalsprache für hierarchische Datenbanken,
Kurzvortrag anläßlich der GI Jahrestagung, Nürnberg, 1977

(49) W. Klutentreter,
Überlegungen zur Portabilität von Datenbankanwendungen,
Aus der wissenschaftlichen Arbeit, GMD Forschungsbericht 1975

(50) P. Koeppe,
Ausbildungsziele, -inhalte und -methoden in der Medizinischen Informatik,
Überarbeitung des Reisensburger Protokolls, Anmerkungen und Literaturhinweise, FU Berlin, 1976, FA 14 der GI und Untergruppe Ausbildungsfragen der AG Systeme und Systementwicklung der GMDS

(51) K.-D. Krägeloh, P.C. Lockemann,
Schichten von Datenmodellen und Datenbanksprachen,
Universität Karlsruhe, ca. 1975

(52) G. Laude,
Entwicklung eines computergestützten Informationssystems für die Materialwirtschaft eines Großklinikums - Fallbeispiel der Medizinischen Hochschule Hannover,
Dissertation TU Hannover, 1978

(53) H.-O. Leilich, G. Stiege, H.CH. Zeidler,
A search processor for data base management systems,
Proceedings 'Very large data bases', Berlin,
13.-15. Sept. 1978

(54) H.-O. Leilich, G. Stiege, H.Ch. Zeidler,
The search processor: A special device for data base management systems,
Informatik Bericht Nr. 7801, Technische Universität Braunschweig

(55) V.Y. Lum, P.J. Owens,
File organization and evaluation modeling system: Users guide,
Final Report, Vol II, Contract AF 30 602-70-c-0814,
IBM Research Laboratory, San Jose, Feb. 1971

(56) F. Nakamura, I. Yoshida, H. Kondo,
A simulation model for data base system performance evaluation,
Proc. AFIPS, 1975

(57) I. Nassi, B. Shneiderman,
Flowchart techniques for structured programming,
SIGPLAN Notices Nr. 8, 1973

(58) G.M. Nijssen,
On the gross architecture for the next generation data base management systems,
Proceedings Information Processing 77 (IFIP), Toronto, 1977

(59) Nicht benannt (anonym)
Systems approach to health care systems analysis,
Medizinische Hochschule Hannover, 1977

(60) W.C. Oney,
Queuing analysis of the scan policy for moving-head disks,
Journal of the ACM, Vol 22, No 3, July 1975

(61) E.A. Ozkarahan, S.A. Schuster, K.C. Sevik,
Performance evaluation of a relational associative processor.
ACM Transactions on data base systems, Vol 2, No 2, June 1977

(62) P.R. Pocklington,
The necessity for, requirements of, and basic design of a general data interpretation and evaluation system (DIES),
In: J. Anderson, J.M. Forsythe, Medinfo 74, Stockholm, 1974

(63) P.L. Reichertz,
Informationssysteme in der Medizin,
IBM Druck, Bonn Bad Godesberg, 1975

(64) P.L. Reichertz,
Medizinische Informatik, Arbeitstagung: Informatik und Informationswissenschaften,
GMD Birlinghoven, April 1976

(65) P.L. Reichertz,
The Medical System Hannover (MSH),
In: N.F. Collen (edtr), Hospital Computer Systems, New York, 1974

(66) P.L. Reichertz,
Medizinische Informatik, Aufgaben, Wege und Bedeutung,
Verhandlungen Gesellsch. Deutscher Naturforscher und Ärzte, Nr. 107, 1973

(67) P.L. Reichertz,
Towards systematization,
Medcomp 77, International congress on computer in medicine, Berlin, Feb. 7-9, 1977

(68) P.L. Reichertz, E. Wolters, R. Engelbrecht,
A teleprocessing interface macro system (TIMS),
Meth. Inf. Med. Nr. 12, 1973

(69) P. Reisner, R.F. Boyle, D.D. Chamberlin
Human factor evaluation of two data base query languages - SQUARE and SEQUEL,
AFIPS National Computer Conference, 1975

(70) T. Risch,
LIDAM, LISP Data Manager,
Datalogilaboriet Report DLU 77/2, Uppsala Univ., 1977

(71) K. Sauter,
Integrierte Datenbank und Patienteninformationssystem Hannover,
Habilitationsschrift, Medizinische Hochschule Hannover, 1972

(72) K. Sauter, P.L. Reichertz, W. Weingarten, B. Schwarz, D. Weigelt,
System controlled structuring and processing tools for an implemented generalized data base management system,
Angewandte Informatik, Nr. 8, 1975

(73) K.Sauter, P.L. Reichertz, W. Weingarten, B. Schwarz,
A system to support high level data description and manipulation of an operational data base system,
Medical Informatics, Vol 1, No 1, 1976

(74) K. Sauter, O. Rienhoff,
Rechnergestützte medizinische Informationssysteme,
Vorlesung Medizinische Hochschule Hannover, WS 1977/78

(75) K.Sauter, P.L. Reichertz, W. Zowe, W. Weingarten, D. Finke, J. Klonk,
Datenbank-gestütztes Patienteninformationssystem für ein Universitätklinikum - Analyse einer sechsjähren Erfahrung,
22. Jahrestagung Deutsche Gesellschaft für Medizinische Dokumentation, Informatik und Statistik,
Göttingen, 1977

(76) K. Sauter, W. Weingarten, D. Finke, J. Klonk, W. Zowe,
Aufbau und Funktion einer zentralen Übersichts- und Verweisdatei in einem Patienteninformationssystem,
Frühjahrstagung 1977 des Fachbereichs Medizinische Informatik der GMDS, Universität Giessen,
31.03. - 01.04.1977

(77) K. Sauter, P.L. Reichertz, W. Weingarten, W. Zowe,
Die Datenbank in einem medizinischen Informationssystem,
Treffen ACM 3/76 des German Chapter of the ACM,
'Praxis des Rechnereinsatzes in der Medizin', München,
25.-26.11.76

(78) E. Schieferdecker,
Neue Methoden und Techniken der Programmierung - Erfahrungen in einem Entwicklungszentrum,
IBM Nachrichten, 26. Jahrg., Heft 232, 1976

(79) H.A. Schmidt,
Architektur und Implementierung von DB-Systemen,
Sommerseminar im Informatikkolleg der GMD,
12.-16.07.1976, GMD Spiegel

(80) P. Schnupp, C. Floyd,
Software, Programmentwicklung und Projektorganisation,
Walter de Gryter Verlag, Berlin, 1976

(81) W. Scholz,
Eine IMS Tuning-Methode,
IBM Nachrichten, 27. Jahrg., Heft 234, 1977

(82) R. Schuberth,
Komplexität - Komplexe Systeme - Komplexe Lösungen,
IBM Nachrichten, Heft 239 und 240, 1978

(83) M.E. Senko, V.Y. Lum, P.J. Owens,
A file organization evaluation model (FOREM),
Proceedings of the IFIP Congress, Edingburgh 1968,
North Holland Publishing Company, 1969

(84) S.W. Shermann, B.S. Brice,
Performance of a data base manager in a virtual memory system,
ACM Transactions on data base systems, Vol 1, No 4,
Dec. 1976

(85) D.B. Shires, H. Wolf (edtrs),
MEDINFO '77,
Proceedings of the second world conference on Medical Informatics, Toronto, 1977

(86) N.C. Shu, et.al.,
EXPRESS: A data extraction, processing and restructuring language,
ACM Transactions on data base systems, Vol 2, No 2,
June 1977

(87) W.A. Spencer,
The health information system (HIS),
In: W. Schneider (managing editor), Computer Programs in Biomedicine, Vol 5, No 3, 1976, North Holland Publishing Company

(88) W. Steinmüller,
Datenschutz bei Risikosystemen - Am Beispiel eines medizinischen Informationssystems,
Nachr. Dok. 28 (1977) Nr. 2

(89) G. Stiege,
Ein neues Verfahren der Datenverarbeitung in Hard- und Software: Suchrechner und Assoziativspeicher,
Informatikseminar Medizinische Hochschule Hannover,
26.11.1975

(90) G. Stiege,
Projektschriften und Dokumentation 'Suchrechner',
Technische Universität Braunschweig

(91) G. Stiege,
Benutzerhandbuch für die Grundsoftware, Suchrechner -
Datenbanksystem,
Technische Universität Braunschweig

(92) S. Stimler,
Leistungsmessung, Leistungsbewertung und Leistungsverbesserung von Datenverarbeitungssystemen,
Oldenbourg Verlag, München, 1976

(93) M. Stonebraker, E. Wong, P. Kreps, G. Held,
The design and implementation of INGRES,
ACM Transactions on data base systems, Vol 1, No 3,
Sept. 1976

(94) L. Svobodova,
Computer performance measurement and evaluation methods,
Elsevier, New York, 1976

(95) K. Terplan,
Computerlast - ihre Beschreibung und Vorhersage,
Angewandte Informatik, Heft 4, 1978

(96) K. Terplan,
Das Effektivitätsdilemma virtueller Systeme,
Angewandte Informatik, Heft 5, 1977

(97) K. Terplan,
Leistungsfähigkeit der Datenverarbeitung -
Steuerung und Methoden der Effektivitätsmessung für Computersysteme,
Verlag Dokumentation, München 1977

(98) H. Wedekind, T. Härder,
Datenbanksysteme I und II,
Reihe Informatik, Nr. 16 und 18, 1974 und 1976,
BI Wissenschaftsverlag

(99) W. Weingarten, J. Klonk, K. Sauter, P.L. Reichertz,
Individual data re trieval by the non-programmer and system supported data manipulation in a complex hierarchically organized data base system,
In: D.B. Shires, H. Wolf (edtrs), MEDINFO '77, Proceedings of the Second World Conference on Medical Informatics,
Toronto, Aug. 8-12, 1977,
North Holland Publishing Company

(100) W. Weingarten, K. Sauter, D. Weigelt, P.L. Reichertz,
The patient information system and its user reactions,
Journees d'informatique medicale 1975, communications des tables rondes, XVI-XV15

(101) W. Weingarten, P.L. Reichertz, K. Sauter, D. Weigelt, W. Zowe,
Patienteninformationssystem für medizinische Daten im MSH
In: S. Koller (Hrsg.). Medizinische Informatik und Statistik, Springer Verlag, Band 1, 1976

(102) N.C. Wilhelm,
A general model of the performance of disk systems,
Journal of the ACM, Vol 24, No 1, Jan. 1977

(103) E. Wolters,
Ein datengesteuertes interaktives System für medizinische Anwendungen (DADIMOPS),
Habilitationsschrift, Medizinische Hochschule Hannover, Nov. 1976

(104) S.B. Yao,
An attribute based model for data base access cost analysis,
ACM Transactions on data base systems, Vol 1, No 1, March 1976

(105) H.Ch. Zeidler,
Zur Entwicklung des Suchprozessors für einen Datenbankrechner,
NTG-Fachberichte, Band 62, VDE Verlag Berlin

(106) W. Zorn,
Wozu Leistungsmessungen an DV-Anlagen.
Informatik-Rechnerabteilung, Universität Karlsruhe,
Interner Bericht Nr. 9, 1976

6. FORMALE SYNTAXDEFINITIONEN

Dokumentation der Kommandos in der PDPL-Sprache

Die Kommandokennung besteht aus den ersten vier Buchstaben. Die folgende Aufstellung ist alphabetisch geordnet.

'Eingabeunterbrechung' soll bedeuten, dass die Programmausführung an dieser Stelle unterbrochen wird, weil eine Eingabe vom Bildschirm (Terminal) bei Dialogbetrieb erwartet wird. Im Stapelbetrieb erfolgt hier automatisch ein Lesevorgang für die Datei SYSIN.

'Protokollierung' bedeutet im Dialogbetrieb die Ausgabe auf dem Bildschirm, bei Stapelbetrieb den Ausdruck über Schnelldrucker.

APTH	Protokoll der Nummer des jetzt gerade aktiven Verarbeitungspfades, für die Fehlersuche.
BCHNno	Bei der seitenweisen Ausgabe von Datenelementinhalten auf Grund der Anweisung ODB(EX,...) kann es im Dialog sinnvoll sein, eine Zeile der Ausgabe nach der Zeilennummer, die jeder Zeile automatisch vorangestellt wird, zu markieren (auszuwählen) und mit den enthaltenen Informationen z.B. einen Direktzugriff auf einen Datensatz in einem anderen Verarbeitungspfad des Auftrages auszuführen (z.B. Einstieg in die Suche über eine Namensliste und weiter bei den Diagnosen des so ausgewählten Patienten). Die Auswahl der Zeile=nr aus einer Ausgabeseite erfolgt mit dem Kommando TAKE,nr. Das Kommando BCHNno speichert vorher für den Fall eines späteren TAKE-Kommandos die Information no=Pfadnummer des Verarbeitungspfades, zu dem auf Grund eines TAKE-Kommandos zu verzweigen ist. NBCH ist gleichbedeutend mit BCHNO und bedeutet das Ausschalten der Verzweigung, d.h. ein TAKE-Kommando hat keine Wirkung mehr.
BCH,nr	Unbedingte Verzweigung der Verarbeitung zum Pfad mit der Nummer=nr des Verarbeitungsauftrages.

BRAK	Eingabeunterbrechung nach Ausgabe von Datenelementinhalten auf Grund einer ODB(EX,...)-Anweisung auf dem Terminal bei Dialogbetrieb oder Drucker bei Stapelbetrieb.
BUFD	Protokoll der Datenpufferbelegung mit Datenelementen durch die DDC-Anweisungen eines Auftrages. Vornehmlich für Systempflege nützlich.
CALL	Der Datensatzzugriff mit Hilfe des über die jeweiligen SFS-Anweisungen eines Datenverarbeitungsauftrages bestimmten Datenverwaltungssystems ist prinzipiell aktiviert und wird bei Durchlauf durch den Verarbeitungspfad ausgeführt. In der Testphase des Auftrags kann es sinnvoll sein, den Datensatzzugriff selbst zu unterdrücken. Dafür dient das Kommando NOCA, das durch das Kommando CALL wieder aufgehoben wird und umgekehrt.
CARD	Die Anzahl der erfolgreichen Datensatzzugriffe wird mitgezählt. CARD protokolliert den augenblicklichen Zählerstand.
CLTRxx	Die Zwischenspeicherung von Suchergebnissen kann in sog. temporären Relationen (Tabellen) erfolgen. Über die Anweisung SFS(nr,TRxx) werden die Relationen mit der Nummer=xx lesend oder auch schreibend angesprochen wie jedes Datenverwaltungssystem in GURS. Die Relationen können mit GET NEXT oder GET UNIQUE gelesen werden. Sie werden auf der Systemdatei von GURS angelegt. Mit dem Kommando INIO werden alle temporären Relationen gelöscht. Mit dem Kommando OPTRxxyy wird die temporäre Relation Nummer=xx mit einer Datensatzlänge=yy Byte für das Beschreiben eröffnet. Mit dem Kommando CLTRxx muss sie nach dem Beschreiben geschlossen werden und steht von da an ohne weitere Öffnungskommandos zum Lesen zur Verfügung (Zwischenergebnisspeicher).
COPY	Bei Dialogverarbeitung kann es sinnvoll sein, die Ausgaben der Anweisung ODB(EX,...) und die der Kommandos APTH, CARD, MESS, RPUT, BUFD, LPUT TIME und STAT auf einem zweiten Gerät, z.B. einer Kugelkopfmaschine zusätzlich zu protokollieren. Mit dem Kommando COPY wird diese Doppelausgabe aktiviert. Das Kommando NCPY hebt COPY auf und umgekehrt. Die Geräteadresse des Zweitgerätes ist mit ad=MHHOOL40 initialisiert, kann aber durch das Kommando CYADad beliebig umgesetzt werden.

COUN	Ein Zähler, der die Datensatzzugriffe jeweils bis 400 hochzählt und dann eine Eingabeunterbrechung erzwingt, soll für das Testen eines Auftrages Endlosschleifen verhindern. Das Kommando NOCO hebt COUN auf und umgekehrt.
CPT1 CPT2	Unbedingte Verzweigung der Verarbeitung innerhalb desselben Pfades zum Programmblock CPT1 bzw. CPT2.
CYADad	siehe ~~BCHNno~~ COPY
DDEFyy	Datenelementdefinitionen sind in der Steuerdatei von GURS gespeichert. Die Korrektur bzw. Neueingabe einer Datenelementbeschreibung zur Datenelementnummer=yy wird über Bildschirmmasken mit diesem Kommando eingeleitet. Die Darstellung des Inhaltes einer Datenelementbeschreibung ohne sie verändern zu wollen, erfolgt auf Grund des Kommandos DSEEyy.
DMSC	Unbedingte Verzweigung der Verarbeitung innerhalb desselben Pfades zum Programmblock CALL (Aufruf der Datenverwaltungssysteme).
DSEEyy	siehe DDEFyy
ENDE	Programmende bzw. Sitzungsende am Bildschirm
IDB1 IDB2	Unbedingte Verzweigung der Verarbeitung innerhalb eines Pfades zum Programmblock mit dem Namen IDB1 bzw. IDB2.
INIO	siehe CLTRxx
IOLO	Protokoll des Inhaltes des Ein/Ausgabebereiches (I/O-Area) als Zeichenkette (String).
LIST	Datenelementinhalte können durch die Anweisung ODB(EX,...) auf dem Terminal bei Dialogbetrieb oder Schnelldrucker bei Stapelbetrieb ausgegeben werden. Es werden zwei Ausgabearten (Aufbereitungsformen) unterschieden: - alle Elementinhalte mit einem Leerzeichen als Zwischenraum in einer Zeile, wobei Oberlängen abgeschnitten werden. Die Ausgabe erfolgt Seitenweise, eine Zeilennummer wird jeder Zeile vorangesetzt. - Datenelementinhalte werden einzeln je Seite ausgegeben, indem noch Texte aus der Datenelementbeschreibung in mehreren Zeilen hinzugefügt werden (Datenelementnummer, Datenelementname, Datenelementbedeutung, u.s.w.).

	I LIST bedeutet die erstgenannte zeilenweise Ausgabe der Datenelementinhalte. Das Kommando NOLI schaltet die Ausgabe um auf die zweitgenannte Art (Einzelausgabe je Seite). Ein beliebiges Hin- und Herschalten während der Ausgabe ist erlaubt.
LPUTnn	I Auf Grund der DDC-Anweisungen wird Platz für Datenelementwerte in einem Datenpuffer reserviert. Wegen der seitenweisen Ausgabemöglichkeit mit LIST besteht dieser Puffer aus mehreren Einzelpuffern, die den Zeilen entsprechend durchnumeriert sind. Mit dem Kommando LPUTnn kann jede Pufferzeile mit der Nummer=nn einzeln in ihrer Gesamtheit protokolliert werden. Vornehmlich Systempflege.
MESSmes	I Protokollierung der Nachricht=mes. Diese Funktion erlaubt z.B. bei nicht gefundenem Datensatz einen Klartext auszugeben.
NBCH	I siehe BCHN
NCPY	I siehe COPY
NOBR	I siehe BRAK
NOCA	I siehe CALL
NOCO	I siehe COUN
NOLI	I siehe LIST
NOPA	I Beim hierarchischen System IMS kann die Bereitstellung eines Datensatzes mit Hilfe des sog. Pathcalls auch so erfolgen, dass alle hierarchisch unmittelbar übergeordneten Datensätze bis zum Wurzelsegment aneinandergefügt im Ein/Ausgabebereich zur Verfügung gestellt werden. GURS hat diese Art des Lesens mit IMS standardmässig vorgegeben, die dadurch verursachten Verschiebungen nach rechts der Datenelemente im Ein/Ausgabebereich werden automatisch berücksichtigt. Das Kommando NOPA schaltet den Pathcall-Lesezugriff aus, dies gilt nur für den Pfad, in dem das Kommando gegeben wurde. Andere Pfade des Auftrages werden also hiervon nicht berührt.

NSPS	Die Ausgabe von Datenelementen über die Anweisung ODB(FI,...) erfolgt in der sequentiellen Datei GURSOUT. Dabei werden die Datenelementwerte aneinander gekettet. Die Datei kann für weitergehende Auswertungen,z.B. auch mit GURS, auf Magnetband gelegt werden. Soll eine weitergehende Auswertung mit SPSS (Statistical Package for Social Sciences) erfolgen, so können die Datenelementbeschreibungen für die Ausgabedatei GURSOUT SPSS-gerecht aufbereitet und mit ausgegeben werden. Um dies zu erreichen, musste vor Auftragsausführung das Kommando SPSS gegeben worden sein. NSPS hebt SPSS auf und umgekehrt.
ODB1 ODB2	Unbedingte Verzweigung der Verarbeitung innerhalb desselben Pfades zum Programmblock mit Namen ODB1 bzw. ODB2.
OPTRx,y	siehe CLTRxx
PTHNoo	Die Datenstruktur, die die Verarbeitung des Auftrages insgesamt steuert, ist die sog. Pfadstruktur. Für die Systempflege ist es sinnvoll, Teile davon zu protokollieren. Um nicht alle Pfade zu protokollieren, wird ein einzelner Pfad mit der Nummer=oo ausgewählt.
QCALno	Verarbeitungsaufträge können, wenn sie einmal auf Grund des Kommandos QUERY dem System GURS übergeben wurden, durch das Kommando QKOR korrigiert werden. Der Auftrag könnte dann mit dem Kommando QTLT der Syntaxanalyse und dem Übersetzer übergeben werden. Falls diese Schritte fehlerfrei waren, kann die Durchführung des Auftrages durch den Interpreter mit dem Kommando START angestossen werden. Der so eingegebene Auftrag ist bei Programm- oder Dialogende verloren, da er nicht automatisch gespeichert wird. Eine permanente Speicherung in der Steuerdatei von Gurs im nicht übersetzten, also PDPL-Format, erfolgt mit dem Kommando QPUTno: no ist dabei die selbst zu vergebende Adresse von 1 bis 999. Das Wiedereinlesen vom Speicher erfolgt mit dem Kommando QGETno. Jetzt könnten die Kommandos QTLT und START erfolgen. Um dies Verfahren abzukürzen, wird die Kommandofolge QGETno, QTLT und START durch das Kommando QCALno automatisch erzeugt und durchgeführt. Auf diese Weise kann ein Auftrag andere Aufträge aufrufen.

QGETno	siehe QCALno
QKOR	siehe QCALno
QPUTno	siehe QCALno
QTLT	siehe QCALno
QUALx	Das Datenmanagementsystem IMS kennt die sog. qualifizierte Suche nach Datensätzen(Segmenten). Das bedeutet, dass z.B. ein Datensatz nicht nur mit dem eigenen Schlüssel für die Suche qualifiziert wird, sondern auch durch Schlüssel der auf dem hierarchischen Pfad darüberliegenden Segmente. Obwohl diese selbst nicht angesprochen werden, kann verlangt werden, dass ihre Schlüssel bestimmte Bedingungen erfüllen. Dieses Verfahren wird als qualifizierte Suche bezeichnet. Voraussetzung ist, dass für den Schlüsselfeldvergleich auch Schlüssel über die Anweisung ODB(SA,...) für die zu qualifizierenden Hierarstufen eingestellt wurden. Da GURS für jede erfolgreiche Suche ein automatisches ODB(SA,...) für das betreffende Segment durchführt, kann das explizite Kommando je nach Lage auch entfallen. x=0 bedeutet, dass für alle Hierarchiestufen eine Qualifizierung der Suche erfolgt, x=j bedeutet, dass nur für die hierarchische Stufe j eine qualifizierte Suche erfolgen soll (Wurzelsegmentstufe=1).
QUER(Y)	siehe QCALno
QU20	Kontrollausgabe der Ergebnisse der Syntaxanalyse. Dieses Kommando dient der Systempflege.
RPUT	Die Ausgabe einer Liste mit dem Kommando LIST erfolgt automatisch, wenn eine Ausgabeseite des jeweiligen Ausgabegerätes voll ist. Sollte bei einer Suche aber vorher das logische Datenbankende erreicht werden, so kann es sein, dass noch einige Zeilen mit Ergebnissen nicht ausgegeben wurden, weil die Seite noch nicht voll war. Das Kommando RPUT veranlasst die Ausgabe solcher Seitenreste, bei Dialogbetrieb auf dem Bildschirm, bei Stapelbetrieb auf dem Drucker.
SALO	Kontrollausgabe der Suchschlüssel, Schlüsseloperatoren, der Suchqualifizierung und des sog. Commandcodes (die letzten beiden Begriffe sind IMS-spezifisch). Das Kommando dient der Systempflege.

SPSS	siehe NSPS
STAR(T)	siehe QCALno
STOP	Unterbrechung der Ausführung eines Verarbeitungsauftrages. Dieses Kommando unterstützt das Austesten von Aufträgen und bewirkt das Anspringen des Fehlerauflaufpunktes. Die Ausführung des Verarbeitungsauftrages bleibt aber aktiv, d.h. ein Rückspringen in die Verarbeitung ist zu jedem Programmblock des aktiven Verarbeitungspfades möglich, aber auch das Wechseln des Pfades mit dem Kommando BCH,nr. Vorher kann man sich Kontrollausgaben, wie z.B. CARD oder APTH geben lassen.
TAKE,nn	siehe BCH,nr
TIME	Protokollierung der Maschinenzeit. Bei Stapelbetrieb kann diese Ausgabe auch auf Lochkarten gestanzt werden, dabei wird noch die Query-Nummer hinzugefügt. Diese Funktion wurde für die Zugriffszeitmessungen verwendet, indem alle 2000 Zugriffe das Kommando TIME ausgeführt wurde.
WHR1 WHR2	Unbedingte Verzweigung der Verarbeitung innerhalb desselben Pfades zum Programmblock WHR1 bzw. WHR2.

PDPL: PROCEDURAL DATA PROCESSING LANGUAGE

<pdpl_task> ::= <DDC_statement><<m>> <path_sequence>

<DDC_statement> ::= DDC(<db_code>,<data_item_name_list>)

<data_item_name_list> ::= <data_item_name> |
<data_item_name_list>
,<data_item_name>

<path_sequence> ::= <path_definition><<1:20>> EOQ

<path_definition> ::= <SFS_statement>

<ACC_statement><<0:1>>

<CMD_statement><<0:n>>

SSA()

<IDB_statement><<0:2>>

<CPT_statement><<0:10>>

<ODB_statement><<0:2>>

<WHR_statement><<0:1>>

<CMD_statement><<0:n>>

CAL()

<SCB_statement><<0:10>>

<IDB_statement><<0:2>>

<WHR_statement><<0:1>>

<CPT_statement><<0:10>>

<ODB_statement><<0:2>>

<CMD_statement><<0:n>>

Bemerkungen:
bitte naechste Seite

Bemerkungen zu Vorseite:

Angabe der Haeufigkeiten von <...> - Ausdruecken erfolgt durch eine angehaengte Doppelklammer <<min. Anzahl, max. Anzahl>>. Eine fehlende Doppelklammer soll bedeuten: <<1,1>>.

m: <<1:40>> Datenelementdefinitionen,
beliebig viele <db_code>.

n: max. 60 <command> je Pfaddefinition

Die mnemonischen Abkuerzungen sind:
DDC data declaration
EOQ end of query
SFS specify segment
ACC access definition
CMD command
SSA segment search argument
IDB into data buffer
CPT compute
ODB out of data buffer
WHR where
CAL call data management
SCB status code command
(nach dem Datensatzzugriff)

<DDC_statement> ::= DDC(<db_code>,<data_item_name_list>)

Bemerkungen:

<db_code> Der Code ist eine zweistellige Abkuerzung fuer Datenbanknamen, Dateinamen oder sonstige Datenverwaltungsschnittstellen des MSH, fuer die in GURS ein Datensatzzugriff vorgesehen wurde. Der Code ist im Daten- und Programmbeschreibungssystem DAPRO gespeichert.
In dieser Arbeit wurden folgende Datenbanken angesprochen:

Code	Datenbankname
CO	CONPAT (Bild 3.7)
LE	LEIPAT (Bild 3.8)
AP	APAT (Bild 3.9)
NA	NAMPAT (Bild 3.10)
CE	CENSUS (Bild 3.10)
AU	AUFPAT (Bild 3.10)
PA	PATFILE(Bild 3.10)

<data_item_name_list> ::= <data_item_name> |
<data_item_name_list>
,<data_item_name>

Die Datenelementnamen muessen gueltige Namen sein, wie sie im Daten- und Programmbeschreibungssystem DAPRO gespeichert sind. Datenbankcode und Datenelementname zusammen sind der eindeutige Name des Elementes im Internen Schema, z.B. (CO,Izahl).

```
<SFS_statement>  ::=  SFS(<no>,<segment_name>)
                       | SFS(<no>,DUMMY)
```

Bemerkungen:

<no> ist eine Nummer aus dem Bereich von 1 bis 20.
Innerhalb der <path sequence> sollen die <no> mit 1 beginnend fortlaufend vergeben werden@ (sonst Fehlergefahr.)
<segment_name> ist ein gueltiger Segmentname (Datensatzname). Die in dieser Arbeit verwendeten Datensaetze mit ihren Datenelementen sind in den Bildern 3.7 bis 3.11 dargestellt. Die Datensatznamen sind im Daten- und Programmbeschreibungssystem (DAPRO) dokumentiert. Fuer das Datenbanksystem IMS sind die Segmentnamen der 'data base definition' (DBD) gleichzeitig auch die Datensatznamen in DAPRO.
DUMMY bewirkt, dass der physische Segmentzugriff unterdrueckt wird, das ACC_statement kann dann entfallen. Alle anderen Statements sind anwendbar.

Beispiele:

```
SFS(1,COROOT)
SFS(6,APARZT)
SFS(2,DUMMY)
```

<ACC_statement> ::= ACC(<function>,<key_qualification>)

<key_qualification> ::= 0 | 1,<key_operator>

Bemerkungen:

<function> sind die in IMS gebraeuchlichen Funktionen
GN, GNP, GHN, GU, GHU, ISRT, REPL, DLET
<key_operator> bewirkt, dass der Schluesselvergleich im
Datenmanagementsystem entsprechend durchgefuehrt wird:
==, >=, >>, <=, << (Die Schreibweise == statt = ist unbedingt einzuhalten).
0 bedeutet, dass kein Schluesselvergleich stattfindet
(unqualifizierter Call).

Beispiele:
ACC(GN,0) ACC(GU,1,==) ACC(GN,1,>=)

<IDB_statement> ::= IDB(<from>,<data_items>)

Bemerkungen:
IDB steht fuer 'Into Data Buffer. Die Anweisung bewirkt, dass aus dem durch <from> bezeichneten Bereich Datenelcmentwerte in den durch die DDC-Anweisung reservierten Plaetze heruebergebracht werden.
<from> bezeichnet, von woher die Datenelementwerte in den Datenpuffer zu transportieren sind.

<from>=<EX> bedeutet Bildschirmeingabe bei Dialogbetrieb und Lesevorgang in der Datei SYSIN bei Stapelbetrieb.
<from>=<FI> bedeutet lesen eines Datensatzes der Datei GURSIN mit einer beliebigen Satzlaenge und Entnahme der angegebenen Datenelemente, von denen angenommen wird, dass sie aneinandergereiht und im externen Format enthalten sind.
<from>=<IO> bedeutet Entnahme der angegebenen Datenelemente aus dem Ein-/Ausgabebereich (i/O-Area), wobei auf Grund der Datenelementbeschreibung internes Format und Position innerhalb des Bereiches bekannt sind.
<from>=<KF> bedeutet die Entnahme der angegebenen Datenelemente aus der sog. 'Key-feed-back-area', in der die Schluesselfelder aller Segmente des hierarchischen Zugriffspfades eines Zielsegmentes von IMS her nach jedem Zugriff gespeichert werden. Die Nicht-IMS-Schnittstellen wurden in GURS so angepasst, dass auch sie dort ihr Schluesselfeld einstellen. Es wird angenommen, dass die Felder im internen Format unmittelbar aneinander gekettet sind.

EX und FI koennen nur vor dem Datensatzzugriff und IO und KF nur nach dem Datensatzzugriff ausgefuehrt werden (Kapitel 2.3.4).

<data_items> stellt eine Nummernliste dar, wobei die Zahlen die Platzziffern der Datenelemente im Datenpuffer sind. Die Platzziffern entsprechen der Reihenfolge der Datenelemente in den DDC statements. So muessen keine Namen angegeben werden (Schreibarbeit), und es koennen Namen mehrfach verwendet werden, um Bufferplatz zu belegen, z.B. (IZAHL,NAME,NAME) mit der zugehoerigen Platznummernliste (1,2,3).

Beispiele:

```
IDB(IO,2,1,4)
IDB(KF,3)
IDB(FI,1,2,3,4,5,6,7,8,9,10)
```

<ODB_statement> ::= ODB(<into>,<data_items>)

Bemerkungen:
ODB = Out of Data Buffer. Die Anweisung bewirkt, dass in den durch <into> bezeichneten Bereich Datenelementwerte aux dem durch die DDC-Anweisung reservierten Plaetze heraus eingestellt werden.

<into> bezeichnet, in wohin die Werte der Datenelemente vom Datenpuffer aus zu transportieren sind.

<into>=<IO> bedeutet Einstellen der angegebenen Datenelementwerte im internen Format (Konvertierung erfolgt automatisch) in den Ein-/Ausgabebereich (I/O-area) an der richtigen Stelle.

<into>=<SA> bedeutet Einstellen der angegebenen Datenelementwerte in den Schluesselbereich im internen Format, direkt aneinandergereiht. Der Schluesselbereich wird benoetigt, um z.B. dem Datenverwaltungssystem den Schluessel fuer einen Direktzugriff zu uebergeben.

<into>=<EX> bedeutet Einstellen der angegebenen Datenelemente im externen Format
- in eine Bildschirmausgabe bei Dialogbetrieb (je nach Aufbereitungsmodus als Liste oder Einzelausgabe, siehe Kommandodokumentation: LIST)
- in die Ausgabedatei SYSPRINT bei Stapelbetrieb (Aufbereitung wie beim Bildschirm, jedoch groesseres Format).

<into>=<FI> bedeutet Einstellen der angegebenen Datenelementwerte im externen Format direkt aneinandergekettet in die Ausgabedatei GURSOUT, deren Satzlaenge und Blockung von GURS auf Grund der Datenelementliste festgelegt wird. Eine Dateibeschreibung mit dem Datenelementaufbau wird im Protokoll mit ausgedruckt.

IO und SA werden nur vor dem Datensatzzugriff und EX und FI nur nach dem Datensatzzugriff ausgefuehrt (Kapitel 2.3.4).

<data_items> analog zu IDB_statement.
dieselbe Platznummer darf mehrfach auftreten.

Beispiele:
ODB(FI,1,3,2)
ODB(SA,7)
ODB(EX,1,2,3,2)
ODB(EX,2,2,2,2,2,2)

```
<WHR_statement> ::= WHR(<log_expression>)
  <log_expression> ::= <log_group> |
                       <log_expression> <or> <log_group>
     <log_group> ::= <data_item> <log_operator>
                     <data_item  |'string'>
                      |<log_group> <and>
                     <data_item> <log_operator>
                     <data_item  |'string'>
```

Bemerkungen:

<data_item> ist eine Zahl (ohne Hochkomma), die die Platznummer des Datenelementes gemaess der Bufferdeklaration (DDC_statements) darstellt, also kein Datenelementname.

```
<or>   ::= |
<and> ::= &
<log_operator> ::=  = | < | > | <= | >= | r= | -> | <-
```

Die Operatoren -> und <- bedeuten 'vor' bzw. 'nach' und werden richtig nur bei Datumsvergleichen im Format TT.MM.JJ durchgefuehrt.

'string': auch Zahlen im logischen Vergleich muessen in der richtigen externen Laenge mit fuehrenden Blanks in Hochkomma eingeschlossen sein, da Zeichenketten ohne Hochkomma als data_item Platzzahl interpretiert werden. Das ergibt moeglicherweise umstaendliche Abfragen, wenn die Zahlen negativ werden koennen.

Klammern im logischen Ausdruck sind nicht erlaubt: die ohne Klammer ermoeglichten Terme entstehen durch logisch richtiges Ausmultiplizieren der Klammern. Abfragen sind gut zu durchdenken, weil z.B. WHR(1r='47110') eine falsche Patientenauskunft dann gibt, wenn '47110' als Ausschlussdiagnose gedacht war, ein Patient aber diese und weitere andere Diagnosen besitzt. Dann naemlich ergeben die anderen Diagnosen bei der datensatzweisen Abfrage ('tupel at a time') fuer diesen Patienten ungewollt Treffer. Durch entsprechend andere Programmierung der Abfrage laesst sich eine Ausschlussdiagnose durchaus ohne besonderen Aufwand erfragen.

Beispiel:

```
WHR(1>2&3r='anton'|3=4|5='  10'|4->5|6<-'01.01.78')
```

```
<CPT_statement> ::= CPT(<data_item> <operator>
                        <data_item | 'string'>)
```

Bemerkungen:

<data_item> analog zu WHR_statement.

<operator> ::= = | * | / | ** | + | - | <> | IN | SU

= ist eine Zuweisung in den richtigen Laengen ueber Teilketten-Funktion (Substring) ohne Konvertierung in Zahlen.

*,/,**,+,- sind arithmetische Operarationen, die mit einer Konvertierung in Gleitkommazahlen verbunden ist. Konversionsfehler werden ueber Fehlercode mitgeteilt, der in einem CMD_statement im Format CMD(CPT,<command>) Aktionen veranlassen kann.

<> bildet die Datumsdifferenz in Tagen aus erstem Element und zweiten Element und ueberschreibt das erste Element im Format F(8). Auch hier sind Fehlercodes im Falle falscher Datumsangaben gesetzt.

IN wendet die sog. Index-Funktion (PL/1) an: Die Funktion prueft, ob das zweite Datenelement als Teilkette im ersten Datenelement enthalten ist. Ist es nicht enthalten, so wird als Ergebnis eine Null rechtsbuendig in das erste Datenelement uebertragen. Ist es enthalten, so wird die Startposition rechtsbuendig uebertragen. Die Anweisung CPT(2IN'PILOT') ergibt fuer das Datenelement den Inhalt ' 4', wenn vorher 'JETPILOT' enthalten war.

SU wendet die sog. SUBSTRING-Funktion (PL/1) an: Die Funktion hat als Ergebnis eine Teilkette. Wenn im Datenelement Nummer 7 die Zeichenkette 'JETPILOT' enthalten ist, so ergibt die Anweisung CPT(7SU'4') als Teilkette 'PILOT', die in das Datenelement Nummer 7 linksbuendig uebertragen wird. Nach vorangegangener Anweisung CPT(2IN'PILOT') des Beispiels fuer die Index-Funktion wird dasselbe Ergebnis durch die Anweisung CPT(7SU2) erzielt.

Beispiele:

```
CPT(1=2)
CPT(1/'10')
CPT(2<>1)
CPT(3<>'01.01.66')
CPT(2IN'PILOT')
CPT(7SU'4')
CPT(7SU2)
```

```
<CMD_statement> ::= CMD(<command>)
                  |CMD(LCC,<nr>,<command>)
                  |CMD(WHR,<nr>,<command>)
                  |CMD(CPT,<command>)
```

Bemerkungen:

CMD-Anweisungen sollen Kommandos ausfuehren. Diese Kommandos werden unbedingt ausgefuehrt, wenn das Format CMD(<command>) vorliegt. Die Ausfuehrung in den zwei anderen Faellen (LCC und WHR sind identisch) ist davon abhaengig, ob bestimmte Bedingungen erfuellt sind:

CPT bedeutet die Ausfuehrung des Kommandos, wenn mindestens eine der CPT-Anweisungen einen Fehlercode gesetzt hat (Programmblock CPT1 oder CPT2, Bild 2.13)

WHR,<nr> bedeutet die Ausfuehrung des <command>, falls in der Pruefung der WHR-Anweisung gemaess <nr> ein 'false' der Bedingung resultiert. Dabei bedeutet <nr>=<0>, dass der gesamte logische Ausdruck 'false' ergibt.
<nr> mit 1<nr<=10 bedeutet, dass die einzelnen durch logisches ODER verbundenen Gruppen <log_group>, durchnumeriert von links nach rechts, jede fuer sich mit ihrem 'false' das ueber <nr> zugeordnete Kommando ausfuehren.
Hierdurch kann z.B. eine vom Datenelementinhalt abhaengige Mehrfachverzweigung von Zugriffspfaden programmiert werden:
z.B. Anweisung WHR(1=r'A'|1=r'B'|1=r'C') in Verbindung mit den Kommandos CMD(WHR,1,BCH,4), CMD(WHR,2,BCH,5) und CMD(WHR,3,BCH,6). Falls das Datenelement mit der Positionsnummer 1 den Inhalt 'B' hat, verzweigt die Verarbeitung zu dem Zugriffs- und Verarbeitungspfad mit der Nummer 5.

<command> Kommandos nach der Dokumentation dieses Kapitels.

Beispiele:

```
CMD(NOLIST)  CMD(BCH,7)  CMD(CPT,STOP)
CMD(WHR,0,IDB1)
CMD(WHR,3,BCH,4)
CMD(WHR,0,TAKE,1)
```

<SCB_statement> ::= SCB(<status_code>,<command>)

Bemerkungen:

<status_code> ::= Leerzeichen | GE | GP | AI | II | GB
| DJ | AO | AC | XX

Der <status_code> gibt den Status nach erfolgtem Datenzugriff durch das Daten-Verwaltungssystem wieder:

Leerzeichen= Zugriff im Sinn der angegebenen Funktion erfolgreich durchgefuehrt.

GE=Datensatz (Segment) nicht gefunden

GP=es wurde kein uebergeordneter Satz vor 'GET NEXT WITHIN PARENT'-Zugriff eingelesen (IMS)

AI=Oeffnungsfehler fuer die Datenbank oder Datei

II=Datensatz schon vorhanden beim Einfuegeversuch

GB=Bei der sequentiellen Suche wurde das Datenbankende erreicht

DJ=Vor Datensatzveraenderung wurde kein Sperraufruf durchgefuehrt (HOLD vor UPDATE)

AO=Unbestimmter Fehler im Datenverwaltungssystem

AD=Falsche Funktion fuer die Datensatzverarbeitung

AC=Hierarchiefehler (IMS)

XX=irgendein Code, der nicht Leerzeichen ist

Beispiele:

```
SCB(GB,STOP)
SCB(XX,BCH,4)
SCB(  ,ODB2)
```

Angabe der mnemonischen Kuerzel fuer den
Suchrechner-Assembler:

BEK Befehl, Erklaere Konstante
BEV Befehl, Erklaere Variable
BID Befehl, Vergleiche auf identisch
BGD Befehl, Vergleiche auf groesser als Datenfeld
BKD Befehl, Vergleiche auf kleiner als Datenfeld
BKN Befehl, Ueberlies Kategorienrest nicht
BKR Befehl, Ueberlies Kategorienrest
BLV Befehl, Lade in den Variablenbereich
BME Befehl, Pruefe unter Maske auf Eins
BMN Befehl, Pruefe unter Maske auf Null
BNL Befehl, Nullbefehl
BSR Befehl, Ueberlies Satzrest
BSN Befehl, Ueberlies Satzrest nicht
VBO Verknuepfung, Boolesche Operation
VFI Verknuepfung, Feldergebnis invertieren
VTE Verknuepfung, Termergebnis bilden
VWK Verknuepfung, Wiederholung innerhalb der Kategorie
VWB Verknuepfung, Boolesche Operation bei Wiederholung
in Kategorie

Syntax des Suchrechner-Assemblers:

<Suchauftrag>

{<Deklaration>}$_{o}^{*}$ {<Normalbefehl>}$_{o}^{*}$
<Kategorienendebefehl> <Satzendebefehl>

<Deklaration>

<Konstantendeklaration>|<Variablendeklaration>

<Konstantendeklaration>

BEK <Sp> <Name> <Sp> <Vergleichswert> : {<Sp>}$_{o}^{*}$

<Sp>

<Zeichenfolge von einem oder mehreren
Leerzeichen>

<Name>

<Zeichenfolge, die mit einem Stern * beginnt,
dem 1 bis 6 Buchstaben oder Ziffern folgen>

<Vergleichswert>

{<Vergleichszeichenreihe>}$_{1}^{*}$<Endekennung>

<Vergleichszeichenreihe>

<Normalzeichenreihe>|<Sedezimalzeichenreihe>|
<Maskierung>

22. 4. 1977

<*Normalzeichenreihe*>

<*Zeichenreihe aus mindestens einem Zeichen des PRIME-Codes, in der keines der folgenden Zeichen enthalten ist:*

Leerzeichen	sp
Pfeil aufwärts	^
Unterstreichung	_
Doppelpunkt	:
Stern	*
Schrägstrich	\

>

<*Sedezimalzeichenreihe*>

^<*gerade, von Null verschiedene Anzahl der folgenden Zeichen:*

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F>^

<*Maskierung*>

\ | ^ \ <*Länge*> ^

<*Endekennung*>

_E | _R

<*Variablendeklaration*>

BEV <*Sp*> <*Name*> <*Sp*> <*Länge*>
<*Endekennung*>:{<*Sp*>}$_{o}^{*}$

<*Länge*>

<*Positive ganze Zahl in Dezimaldarstellung, ohne Vorzeichen, ohne führende Nullen*>

<Normalbefehl>

<Suchbefehl>|<Nullbefehl>|
<Kategorienendebefehl>|<Ladebefehl>

<Suchbefehl>

<Suchoperation> {,<Verknüpfungscode>}$_{0}^{5}$
{<Sp>}$_{0}^{*}$ <Vergleichsangabe>:{<Sp>}$_{0}^{*}$

<Suchoperation>

BID | BGD | BKD | BME | BMN

<Verknüpfungscode>

VFI | VBO | VTE | VWB | VWK

<Vergleichsangabe>

<Vergleichswert>|<Adressenangabe>

<Adressenangabe>

<Name>{+<Länge>}$_{0}^{1}$

<Nullbefehl>

BNL{*<Länge>}$_{0}^{1}$ {,<Verknüpfungscode>}$_{0}^{5}$:{<Sp>}$_{0}^{*}$

<Kategorienendebefehl>

<Kategoriencode>{*<Länge>}$_{0}^{1}$
{,<Verknüpfungscode>}$_{0}^{5}$:{<Sp>}$_{0}^{*}$

<Kategoriencode>

BKN | BKR

<Ladebefehl>

BLV{,<Verknüpfungscode>}$_{o}^{5}$<Sp>
<Adressenangabe>:{<Sp>}$_{o}^{*}$

<Satzendebefehl>

BSN | BSR

LEBENSLAUF

Werner WEINGARTEN, geboren am 11. Dezember 1942 in Hamburg.

Eltern: Wilhelm Weingarten und Ehefrau Alwine geb. Suhr.

Schulbesuch	:	1949 - 1955	Volks- und Mittelschule Bassum
		1955 - 1963	Neusprachliches und mathematisch-naturwissenschaftliches Gymnasium Graf Friedrich-Schule in Diepholz Reifeprüfung Ostern 1963
Praktische Ausbildung	:	1963 - 1966	Wehrdienst und Offizier auf Zeit, 6 Monate Industriepraktikum vor dem Studium
Studium	:	1966 - 1971	Elektrotechnik (Nachrichtenverarbeitung) an der Technischen Universität Hannover Diplomurkunde Dezember 1971
Berufstätigkeit:		Dez. 1971	Verwalter einer wissenschaftlichen Assistentenstelle und
		Mai 1972	Wissenschaftlicher Assistent im Department Biometrie und Medizinische Informatik der Medizinischen Hochschule Hannover (Prof. Dr. Reichertz, Prof. Dr. Sauter)
		Okt. 1978	Dezernent für Grundsatzfragen Datenverarbeitung, Niedersächsisches Landesverwaltungsamt Hannover
		Mai 1980	Technischer Referent beim Niedersächsischen Minister des Innern