TELEFUNKEN
COMPUTER

DBS
Benutzerbeschreibung

SYSTEM TR 440

DBS

Datenbanksystem für den TR 440

Benutzerbeschreibung

Einführung in Datenbanken und Informationssysteme 1

Datenbankorganisation 2

Speicherorganisation und integrierte Datenhaltung 3

DBS-Programmierung 4

DBS-Übersetzer 5

DBS-Formatierer 6

Fehlerausgänge 7

DBS-Dialogsprache 8

Anhang 9

Rückfragen zu dieser Schrift,
Berichtigungen und Ergänzungen
bitte an
TC/VK
Walter, Tel. 3 10 26

Änderungsstand

1	2														

In diese Felder sind fortlaufend die Nummern der jeweiligen Änderung einzutragen. Sie können dadurch feststellen, ob Sie von uns alle Änderungen erhalten haben.

Beispiel:

1	2	3	4	5			

Sie haben die fortlaufenden Nummern 1 bis 5 unserer Änderungen eingetragen. Bekommen Sie jetzt die Änderung mit der fortlaufenden Nummer 7, so wissen Sie, daß Ihnen die Änderung Nummer 6 fehlt. Anforderungen bezüglich des Änderungsdienstes richten Sie bitte an die untenstehende Adresse, Abt. TC/VS13.

TELEFUNKEN COMPUTER GMBH
775 Konstanz
Max-Stromeyer-Str. 116

Bestell-Nr.: 440.G2.03
Ausgabe: 0273-TC/VK/Wa

Mai 72

INHALT

VORBEMERKUNGEN

1. EINFÜHRUNG IN DATENBANKEN UND INFORMATIONSSYTEME 1

2. DATENBANKORGANISATION 1

2.1. Rechnerunabhängige Datenhaltung und Anwendungen 1
2.2. Gebietskonzept und betriebliche Organisation 1
2.3. Seitenkonzept 1
2.4. Integrierte Datenhaltung 2
2.5. Strukturdefinition 3
2.6. Datenbank-Organigramm 4
2.6.1. Darstellung der Daten (Satztypen) 4
2.6.2. Darstellung der Beziehungen 4
2.6.3. Hierarchien 5
2.6.4. Beispiel 5
2.7. Datenbankbeschreibung und Datenmanipulation 6

3. SPEICHERORGANISATION UND INTEGRIERTE DATENHALTUNG 1

3.1. Speicherorganisation 1
3.1.1. Datenbankträger 1
3.1.2. Elemente der Datenbankspeicherorganisation 1
3.1.3. Gebietsorganisation 2
3.1.4. Seitenorganisation 3
3.1.5. Datensätze 8
3.1.6. Indexsätze 9
3.1.7. Adressen 9
3.1.8. Kettenfelder 9
3.1.9. Datenteil 10
3.2. Speicherungs- und Verarbeitungsformen 10
3.2.1. Random (RAN) 10
3.2.2. DIREKT (DIR) 10
3.2.3. NAHE 11
3.2.4. SEQUENTIELL (SEQ) 11
3.2.5. INDEX-SEQUENTIELL (ISQ) 11
3.3. Die Datenintegration 12
3.3.1. Kettentechniken 12
3.3.2. Die Kettenordnungen 13
3.3.3. Kettentabellen 15

4. DBS-PROGRAMMIERUNG 1

4.1. Sprachelemente 1
4.1.1. Übersicht der Systemteile von DBS 1
4.1.2. DBS-Programmiersprache 1
4.1.3. DBS-Übersetzer 1
4.1.4. DBS-Formatierer 1
4.2. Datenbankbeschreibungssprache (DBB) 2
4.2.1. Formaler Aufbau der DBS-Parameter 2
4.2.2. Gliederung der DBS-Parameter 2
4.2.3. Gebietsparameter 3
4.2.4. Datenparameter 3
4.2.5. Strukturparameter 4
4.2.6. DBS-Parameter (Übersicht) 4
4.2.7. DBS-Parameter 6
4.3. Datenmanipulationssprache (DMS) 21
4.3.1. Datenbankprozessor 21
4.3.2. Mnemotechnische Befehlscodes 25
4.4. DBS-Speicherplan 25

4.4.1. Nachrichtenvermittlungsblock (NVB) 26
4.4.2. Stellvertretersatz (STV) 27
4.5. Argumentlisten 27
4.5.1. DBS/TAS 28
4.5.2. DBS/COBOL 28
4.5.3. DBS/ALGOL 28
4.5.4. DBS/FORTRAN 28
4.6. DBS-Befehle 28

5. DBS-ÜEBERSETZER 1

6. DBS-FORMATIERER 1

7. FEHLERAUSGAENGE 1

7.1. Fehlerbehandlung 1
7.2. Fehlercodes für die DMS 1
7.3. Fehlercodes im Input/Output-Controller 1
7.4. Fehlercodes der DBS-Utilities 2
7.5. Fehlermatrix der Datenmanipulationssprache 3

8. DBS-DIALOGSPRACHE 1

8.1. Der Aufbau der Dialogsprache 1
8.2. Erstellen der DBS-Datenbank 2
8.3. Der Systemsteuerbereich 2
8.4. Allgemeine Form einer Anfrage 2
8.5. Einfache Anfragen 3
8.6. SUCHEN, Lokalisierung eines Satzes oder einer Kette 3
8.6.1. Komplexe Kettenstrukturen 6
8.7. WENN-Klausel 7
8.8. Die Anweisungen der Dialogsprache 8
8.8.1. AENDERN 8
8.8.2. AUSGEBEN 10
8.8.3. SUMME 11
8.8.4. ZSUM 12
8.8.5. ZAEHLEN 12
8.8.6. RECHNEN 12
8.8.7. DURCHSCHNITT 12
8.8.8. INFORMIEREN 13
8.8.9. LISTE 13
8.8.10. ENDE 13
8.8.11. Übersicht über die Syntax der Dialogsprache 14
8.8.12. Fehlermeldungen 16

9. ANHANG 1

9.1. Maschinenausstattung 1
9.2. DBS-Utilities 2
9.2.1. DBS-DUMP 2
9.2.2. Testparameter 2
8.3. Glossary 3

VORBEMERKUNGEN

DBS 440-Datenbanksystem für das Rechensystem TR 440- ist ein Produkt der Anwendungsentwicklung bei AEG-TELEFUNKEN.

Die Systemplanung von DBS 440 wurde von der Anwendungsentwicklung im Jahre 1969 nach umfangreichen Analysen und Vergleichen der auf dem Markt befindlichen Konkurrenz-Systeme und der CODASYL-Vorschläge (CODASYL = Conference of data system languages) aufgenommen. Für die Erstellung von DBS 440 wurden drei Ausbaustufen festgelegt.

Die Systemerstellung der ersten Ausbaustufe von DBS 440 wurde im Februar 1970 abgeschlossen. Ziel dieser Ausbaustufe war die Erstellung der Kernteile der DBS 440-Makrobefehle für den Datenverkehr zwischen DBS-Anwenderprogrammen (Physischer E/A-Modul) und der Datenbank auf der Grundlage der direkten, der starren und logisch sequentiellen und der index-sequentiellen Speicherungs- und Verarbeitungsformen.

Ziel der zweiten Ausbaustufe, die im März 1970 begonnen wurde, war die Erstellung des DBS-Übersetzers, die automatische Adreßverkettung, der Einbau der Speicherungsform Random, die Erstellung der assoziativen Makros und der Anschluß von DBS an COBOL und ALGOL. Mit der zweiten Ausbaustufe von DBS 440 werden die Forderungen der Anwender an DBS 440 in militärischen und zivilen Führungssystemen sowie in Informationsbereitstellungssystemen an eine leistungsfähige integrierte Datenhaltung abgedeckt.

Für das bessere Verständnis dieses Handbuches werden folgende Unterlagen zur Lektüre empfohlen:

- TR 440 Kommandohandbuch
- TR 440 TAS-Handbuch
- TR 440 COBOL-Handbuch
- TR 440 Systemdienste BS3

Dieses Handbuch ist für den Gebrauch durch Organisatoren und Programmierer geschrieben. Für Anregungen und Kritiken an Form, Inhalt und Benutzbarkeit dieses Handbuches sind wir dankbar.

1. EINFÜHRUNG IN DATENBANKEN UND INFORMATIONSSYSTEME

Der Einsatz von Informationssystemen ist hauptsächlich durch den wachsenden Informationsbedarf notwendig geworden, der nicht zuletzt durch die wirtschaftliche und technische Entwicklung bedingt ist. Diesem Informationsbedarf steht ein "Überangebot" an Informationen gegenüber, und das Hauptproblem ist nicht, diese Informationen lediglich zu sammeln, sondern aus der Menge der Informationen die für den jeweiligen Informationssuchenden relevanten herauszufinden. Betrachtet man z.B. den betriebswirtschaftlichen Bereich, so erkennt man, daß der Erfolg der Unternehmensführung aufgrund der zunehmenden Komplexität der wirtschaftlichen und technologischen Entwicklung und des ebenfalls zunehmenden Konkurrenzkampfes stark von der Qualität und nicht von der Quantität der Informationen für die Entscheidungsfindung abhängt und somit das eigentliche Führungsproblem geworden ist.

Aus der pragmatischen Perspektive betrachtet, sind Informationssysteme als Datenorganisations- und Datenaufbereitungsverfahren auf Rechenanlagen definierbar, die bestimmte Vorgänge und Abläufe in Kommunikationsprozessen unterstützen. Diese Vorgänge und Abläufe müssen, damit sie überhaupt maschineller Bearbeitung zugänglich sind, formalisier- und damit automatisierbar sein.

Unter einem Kommunikationsprozeß wird hier grundsätzlich ein Vorgang verstanden, in dem Nachrichten zwischen Sender und Empfänger ausgetauscht werden. Kommunikationspartner, also Sender und Empfänger, sind in einem Informationssystem vorzugsweise Mensch und Maschine.

Es gibt Anwendungsbereiche, in denen die Geschwindigkeit des Kommunikationsvorganges eine entscheidende Rolle spielt. Stellvertretend für solche Informationssysteme sind die Flugsicherungssysteme. Bei diesen Systemen werden - bedingt durch die von der Aufgabe her geforderte kurze Reaktionszeit - die Daten im Kernspeicher der Datenverarbeitungsanlage gehalten, so daß Eingabe, Auswertung und Ausgabe nahezu ohne Zeitverlust erfolgen.

Eine Auslagerung der Daten aus dem Kernspeicher auf einen Sekundärspeicher und die Organisation der Daten als Datenbank ist für diesen Anwendungsfall nicht die Norm. Man bezeichnet solche Systeme auch als direkte Informationssysteme.

Bei den meisten Informationssystemen nimmt jedoch das Datenvolumen ein so großes Ausmaß an, daß dieses nicht mehr im Kernspeicher gehalten werden kann, sondern auf externe Speichermedien ausgelagert werden muß. Beispiele solcher Systeme sind auf der einen Seite Management-Informations-Systeme (MIS), Führungssysteme für industrielle, militärische, verwaltungstechnische oder wissenschaftliche Zwecke und die Dokumentationssysteme (Informationsbereitstellungssysteme = IBS) mit ihrem breiten Anwendungsspektrum auf der anderen Seite. Bei all diesen Systemen ist die Menge der zu verwaltenden Daten bzw. die Vielfalt der Möglichkeiten, unter denen die Daten ausgewertet und manipuliert werden können, kritischer als die Geschwindigkeit der Auskunftserteilung und Informationsauswertung.

Das Auslagern dieser Daten auf periphere Speicher und das relativ schnelle Bereitstellen einer Teilmenge dieser Daten zum Zweck spezieller Auswertungen ist erst durch die neuere Hard- und Softwareentwicklung möglich geworden. Das klassische Konzept der bandorientierten EDV-Systeme führte die Stammdaten und die zugehörigen Bewegungen (Änderungen) eines Organisationsbereiches ablaufbezogen in dezentralen Dateien.

Die Aufteilung des Gesamtvolumens in eine Vielzahl funktioneller Einzeldateien (Stammbänder) erfolgte im wesentlichen nach dem Kriterium der Auswertbarkeit, mit dem Erfolg, daß viele Daten im System mehrfach gehalten wurden und dadurch die Datenredundanz und der damit verbundene aufwendige Änderungsdienst sich nicht selten zum systembelastenden Faktor entwickelte. Die Beschreibung der Datengruppen und -elemente erfolgte ausschließlich in den einzelnen Programmen, die diese Dateien verarbeiten.

Durch den Einsatz von Magnetplatten- und Trommelspeichern kam gegenüber der sequentiellen Magnetbandspeicherung der diesen Medien spezifische Vorteil des Direktzugriffs hinzu. Diese Neuerung war ein entscheidender Schritt auf dem Weg zur angestrebten Datenintegration. Mittels geeigneter Organisationsformen können die multiplen funktionell organisierten Einzeldateien eines Unternehmens bzw. einer Organisation zu einem integrierten Datenbestand zusammengefaßt werden. Eine solche Zusammenfassung bezeichnet man als Datenbank. Unter einem Datenbanksystem wird demgemäß ein System verstanden, das zum Führen und Verwalten von großen Datenmengen auf Direktzugriffsspeichern eingesetzt wird.

Wesentliches Merkmal dabei ist, daß das Kriterium für den Aufbau dieser Datenbank sich nach den Daten und ihren Strukturen richtet und die spätere funktionelle Auswertbarkeit der Daten eine untergeordnete Rolle spielt. Nur eine solche datenbezogene statt funktionsbezogene Organisation macht es möglich, daß universelle und vielschichtige Auswertungen dieser Daten vorgenommen werden können.

Ein Informationssystem (IS) besteht aus den Komponenten Hardware, Anwendungssoftware und Betriebsorganisation. Diese Komponenten werden im wesentlichen von den Anforderungen bestimmt, die an das IS als Informationsversorgungssystem (vorzugsweise für die Entscheidungsfindung) und als Rechenleistungsversorgungssystem (Rechenzentrumsbetrieb) gestellt werden.

Die Hardware - der zentrale Rechner, die der Kommunikation dienenden Datennetze und Endstellengeräte (Fernschreiber, Sichtgeräte, Datenstationen) und die Speicher für den Datenbankbetrieb - ermöglicht seit langem aufgrund der ausgereiften Grundprogramme für den Teilnehmer- und Teilhaberbetrieb den Aufbau von speziellen Anwendungen wie auch von allgemeinen Informationssystemen.

Die größten Schwierigkeiten treten ohne Zweifel bei der Organisation eines dynamischen Unternehmens auf, hier insbesondere bei der Organisation des Informationsflusses (Datenerfassung, Datenaufbereitung und Speicherung) und bei dem Aufbau rechnerunabhängiger datenbankorientierter Anwendungen.

Die Anwendungssoftware eines rechnerunabhängigen Informationssystems besteht organisatorisch aus folgenden Bausteinen:

1. Systemsteuerung (Monitor)
2. Problemlösende Anwendungsprogramme (Logische Module)
3. Datenbank

Die Logischen Module sind für den jeweiligen Anwendungsfall zu erstellen. Der Monitor und das Datenbanksystem dagegen können aus der Standardversion entsprechend den Anforderungen des Anwenders generiert werden.

Der Monitor steuert den Ablauf der Logischen Module, er selbst wird von dem Betriebssystem des jeweiligen eingesetzten Rechners gesteuert (vgl. Bild 1.1).

Die Datenbank ist die Basis des IS. Die mit Hilfe der Datenbank erzielte Datenintegration kann als ein Baustein innerhalb eines integrierten Informationssystems betrachtet werden.

Es ist leicht zu erkennen, daß für eine derartige Datenorganisation umfangreiche Softwareleistungen benötigt werden, die diese komplexen Probleme organisatorischer, datentechnischer und methodischer Art bewältigen.

Informationssystem

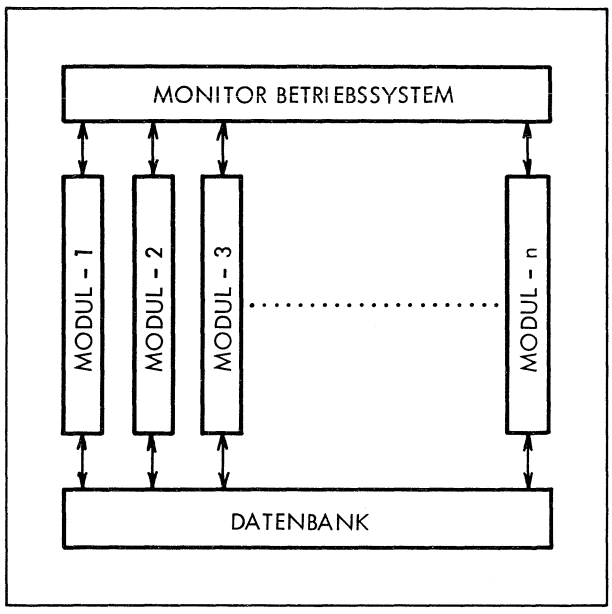


Bild 1.1 Softwarekomponenten eines IS

2.1. Rechnerunabhängige Datenhaltung und Anwendungen

Der Aufbau einer Datenbank ist ein langwieriger und kostenintensiver Prozeß. Neben einer umfangreichen betrieblichen Organisation des Datenflusses (Wo entstehen wann welche Daten? Wer braucht welche Daten in welcher Form, zu welcher Zeit und wofür?) treten die noch immer sehr lohnintensive Datenerfassung und die nur unter der Verantwortung von qualifizierten Spezialisten (Data Manager, Data Administrator) stehende Datenhaltung von Einzeldateien und ganzen Datenbanken auf.

Der Aufbau der Datenbank eines Informationsbereitstellungssystems für die Literaturdokumentation dauert erfahrungsgemäß mindestens fünf Jahre. Dabei sind Informationsbereitstellungssysteme im Vergleich zu Führungssystemen die systemtechnisch einfacher aufgebauten Informationssysteme (IS). Der Aufwand für den Aufbau der Datenbank eines industriellen Führungssystems ist natürlich abhängig von der Komplexität des jeweiligen Unternehmens. Die Aufbauarbeiten werden aber kaum in einer Zeit unter sieben Jahren zu leisten sein. Fachberater Linguisten, Systemberater und Programmierer müssen hart und zielstrebig an den jeweiligen organisatorischen und programmiertechnischen Anwendungsproblemen arbeiten, bis sich der erste Erfolg einstellt.

Für den Aufbau eines Informationssystems sind fünf Jahre eine kurze Zeit. In dieser Zeit kann der Anwender von einem Computer-Generationswechsel überrascht werden. Hohe Aufwendungen entstehen dann bei einem Austausch von nichtkompatiblen Anlagen, falls die Anwendungen, also auch das IS, nicht rechnerunabhängig konzipiert und auch programmiert sind.

Die auf der Datenbank laufenden Anwendungsprogramme unterliegen einem ständigen Änderungsdienst, der verursacht wird durch organisatorische Änderungen im Betrieb, durch Einführung neuer Verfahren usw. Es liegt daher im Interesse des EDV-Anwenders, mit seinen Programmen möglichst rechnerunabhängig zu sein. Rechnerunabhängige Programme können leichter dokumentiert und gewartet werden, sie können bei einer Umstellung aufgrund eines Maschinenwechsels weitgehendst übernommen werden.

Dem allgemeinen organisatorischen Aufbau der Bibliothek, des Unternehmens, der Behörde usw. muß der Aufbau des Informationssystems entsprechen, insbesondere der kontinuierliche Aufbau der Datenbank und der auf ihr laufenden Anwendungssysteme (IBS, MIS).

DBS ist ein Weg für die rechnerunabhängige, problemorientierte Organisation der Daten. Die Anwendungen können in der Programmiersprache COBOL geschrieben werden. Der Anschluß an ALGOL und FORTRAN ist für eine weitere Ausbaustufe vorgesehen.

2.2. Gebietskonzept und betriebliche Organisation

Die Datenbank ist eine Zusammenfassung von Datenbeständen eines oder mehrerer abgeschlossener Organisationsbereiche. Jeder Datenbestand ist ein Komplex von Datenelementen eines bestimmten Typs und hat ein eigenes Format und eine eigene Struktur.

Die DBS-organisierte Datenbank kann nach Sachgebieten gegliedert werden. Sämtliche Daten eines Sachgebietes werden in einem Datenbankspeichergebiet geführt.

Jedes Gebiet kann in Bereiche unterteilt werden. Das Sachgebiet VERKEHR z.B. kann in die Bereiche STRASSENVERKEHR, LUFTVERKEHR und SCHIFFAHRT eingeteilt werden.

Die Speichergebiete einer Datenbank werden analog zu den entsprechenden Organisationsgebieten des Anwenders angelegt und gegliedert. Die Zahl der Gebiete einer Datenbank und die Zahl der Bereiche je Gebiet sind nicht begrenzt. Als Beispiel wird im Bild 2.1 die mögliche Datenbankaufteilung einer Bibliothek wiedergegeben.

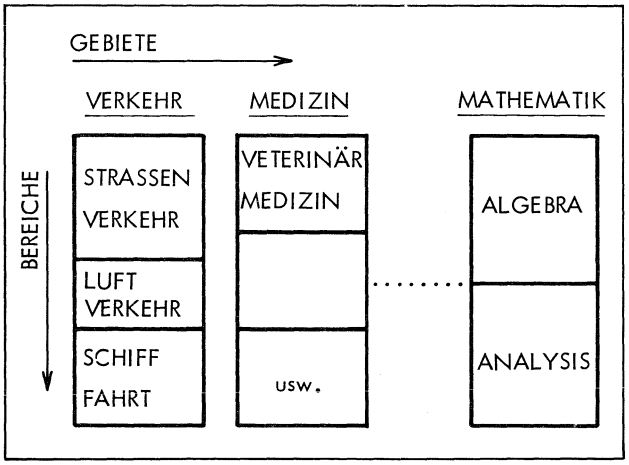


Bild 2.1 Datenbank und Gebietskonzept

2.3. Seitenkonzept

Jedes Gebiet besteht aus System- und Datenbereichen. Die Datenbereiche eines Gebietes werden in Seiten eingeteilt. Die Seiten eines Gebietes haben die gleiche Länge; die Seitenlänge kann von Gebiet zu Gebiet verschieden sein. Die Seiten eines Gebietes werden aufsteigend durchnumeriert, beginnend mit der Seitennummer 1.

2.4. Integrierte Datenhaltung

Um die Reaktion der Datenbank und damit die Qualität des darauf aufbauenden Informationssystems zu erhöhen, werden die Daten integriert gespeichert, d.h. logisch zusammenhängende Daten werden nach Möglichkeit physikalisch in demselben Speicherblock (Seite) abgelegt, damit sie durch nur einen Datentransportvorgang für die weitere Bearbeitung zur Verfügung gestellt werden können.

Die Zugriffszeiten der Datenbankspeicher sind im Verhältnis zur Bearbeitungszeit sehr groß, sie betragen z.B. beim Wechselplattenspeicher WSP 414 im Durchschnitt 45 ms, die Verarbeitungszeit dagegen ist aufgrund der hohen internen Rechenleistung des Großrechners TR 440 klein. Integrierte Datenspeicherung und kurze Reaktiongszeiten sind Grundforderungen an ein modernes rechnerunabhängiges Datenbanksystem.

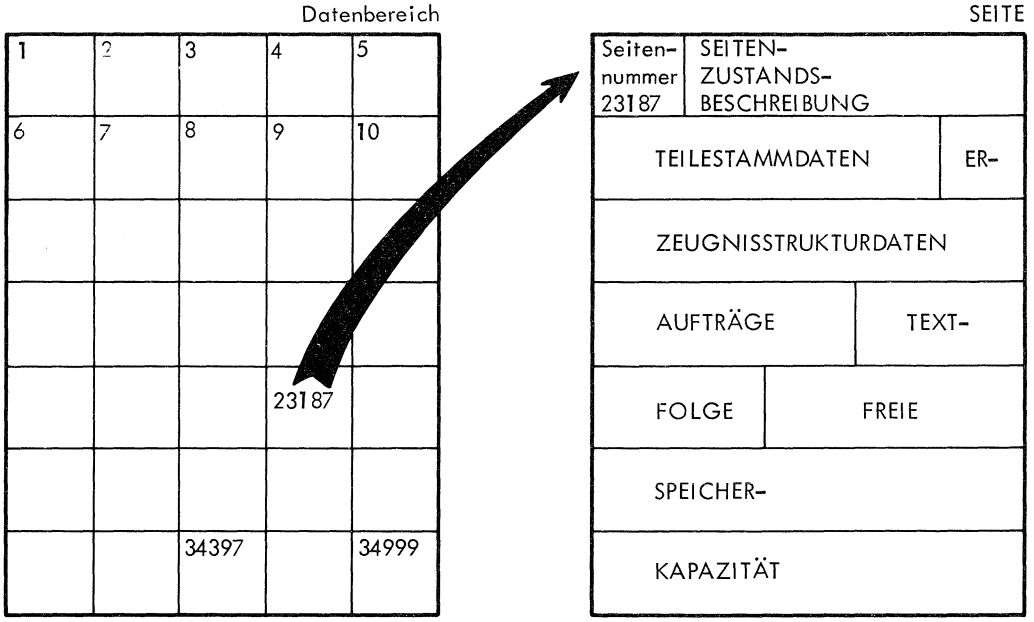


Bild 2.2 Seitenkonzept

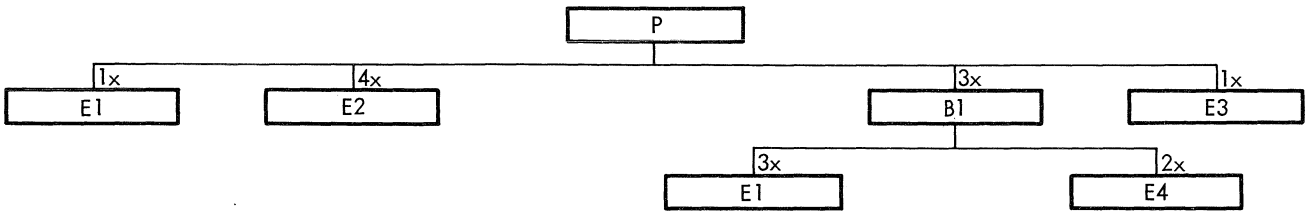
Allgemeine Beschreibung der Daten

In die Speichergebiete der Datenbank werden Datenelemente geladen. Die Datenelemente können beliebigen Typs (Stücklisten; Teilestamm- und Erzeugnisstrukturdaten; Arbeitspläne, Aufträge; Patentunterlagen usw.) und beliebigen Formats (fester Aufbau und feste Länge der Datenfelder, feste Feldfolge eines Datensatzes; variabel im Aufbau und Länge) sein. Die Zahl der Datenelemente ist beliebig.

Für die Abschätzung des Datenvolumens eines Bereiches, eines Gebietes bzw. einer Datenbank, insbesondere aber für die Datenbankprogramme (Datenerfassung, Auswertung, Speicherung und Weiterleitung) ist die Festlegung des Datenaufbaues unbedingt Voraussetzung.

Die Datenbank eines Unternehmens bestehe z.B. aus den Gebieten UNTERNEHMENSLEITUNG, FERTIGUNG, VERTRIEB und VERWALTUNG. Das Gebiet FERTIGUNG wird gegliedert in die Bereiche STUECKLISTEN, ARBEITSPLAENE usw.

Produktstruktur



BAUKASTENSTÜCKLISTE

TEILENUMMER : 523

BENENNUNG : P

POS	MENGE	TEILENUMMER	BENENNUNG
01	1	770	E1
02	4	1583	E2
03	3	1020	B1
04	1	1740	E3

Bild 2.3 Produktstruktur und Stückliste

Teilestammdaten

TEILENUMMER	BENENNUNG	DIS	BESTAND

Erzeugnisstrukturdaten

POS	MENGE

Bild 2.4 Feldaufbau und Feldfolge von Teilestamm- und Erzeugnisstrukturdaten

Unter "Teil" wird ein Endprodukt (P), Baugruppe (B), Einzelteile usw. also allgemein ein Erzeugnis verstanden. Für jedes Teil wird nur ein einziger Teilestammsatz (TST) und für jede Stücklistenposition ein Erzeugnisstruktursatz (EST) angelegt.

Bild 2.3 zeigt als Beispiel für die Beschreibung von Daten eine Produktstruktur und die daraus resultierende Stückliste. Bild 2.4 befaßt sich mit dem Feldaufbau der beschreibenden Sätze.

Jeder DBS-organisierte Datensatz besteht aus zwei Teilen:

Der Datenteil besteht aus Datenfeldern. Aufbau und Reihenfolge der Datenfelder bestimmt der DBS-Anwender. Der Steuerteil wird vom DBS automatisch erstellt, im Steuerteil wird u.a. der Satztyp geführt.

Satztypen und Klassenbildung

Werden nun die Teilestamm- und Erzeugnisstrukturdaten einer Stückliste und die Arbeitspläne zusammen mit dem zugehörigen Primärbedarf (Kundenaufträge für das betreffende Erzeugnis) und den Fertigungsaufträgen in einer Seite gepackt gespeichert, dann muß der Datenbankprozessor jederzeit an einem Code erkennen, von welcher Art (Typ und Format) das Datenelement ist, das er z.Z. bearbeitet. Denn um z.B. den Lagerbestand in einem Teilestammsatz (TST) des Schlagbohrers mit der Teilenummer SB 2-400 aufgrund von Lagerzugängen zu ändern, muß der Prozessor aus einer Folge von beliebigen Datensätzen in einer Seite den bestimmten angeforderten TST auffinden.

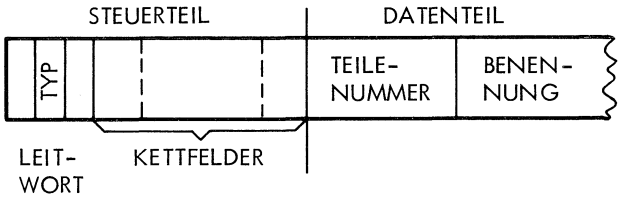


Bild 2.5 Aufbau des Datensatzes

Für die integrierte Datenorganisation wird eine Klassenbildung von Daten eingeführt, und zwar für die allgemeine Organisation der Datenbestände und für die Steuerung und Beschleunigung des Eingabe/Ausgabe-Verkehrs zwischen dem zentralen Arbeitsspeicher und dem Datenbankspeicher. Die Klasse, zu der ein Satz gehört, wird durch den sog. Satztyp festgelegt. Der Satztyp ist ein Zahlencode. Er wird bei der Datenbeschreibung durch den DBS-Parameter *SATZTYP= ... definiert und verschlüsselt im Steuerteil des Datensatzes mitgeführt.
Das Gebietskonzept und die Klassenbildung sind Voraussetzungen für die allgemeine und automatische Organisation von Datenbeständen.

2.5. Strukturdefinition

Strukturen sind Beziehungen (Relationen) zwischen je zwei Datenelementen. Unter Strukturdefinition wird die Beschreibung des Aufbaus einer Datenbank verstanden.

DBS-Ketten

Für DBS werden die Daten in sofort auswertbaren Strukturen (Zusammenfassung logisch zusammenhängender Sätze nach Auswertungsgesichtspunkten) organisiert. Das Hilfsmittel für die speicherungstechnische Organisation derartiger Daten ist die Kette. In einer Kette werden Sätze, die zueinander in irgendeiner Beziehung stehen, zusammengefaßt. Eine Kette solcher Sätze entsteht, wenn in jedem Satz einer derartigen Gruppe die Adresse des nächstfolgenden Satzes dieser Gruppe gespeichert wird. Im letzten Satz steht das Kettenendezeichen. Der erste Satz der Kette heißt ANKER, sämtliche sonstige Sätze der Kette heißen GLIEDER. Die Kette erhält einen Namen, durch ihn wird sie identifiziert (s. Bild 2.7).

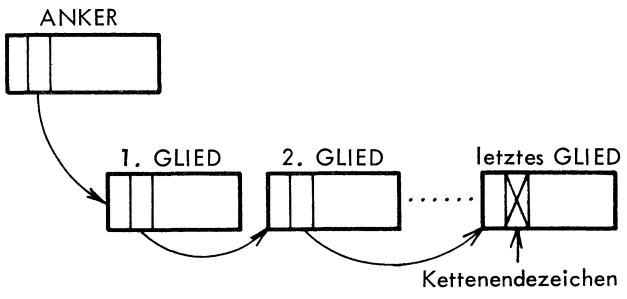


Bild 2.7 Kette

Kettengesetze

Für DBS-Ketten gelten folgende Gesetze:

1. Jede Kette enthält stets genau nur einen Anker. Im Anker steht die Adresse des ersten Gliedes in dieser Kette, d.h. der Anker ist stets der erste Satz der Kette.

2. Jede Kette kann neben dem Anker jede beliebige Anzahl von Gliedern enthalten. Jedesmal, wenn ein Glied in eine Kette eingespeichert wird, wird in dem Kettenfeld des Vorgängers die Adresse des neu eingespeicherten Gliedes eingesetzt; dieses neu eingespeicherte Kettenglied verweist auf seinen Nachfolger, falls es nicht die Endstelle der Kette ist.

3. Jedesmal, wenn ein Anker gespeichert wird, wird eine Kette angelegt, wenn ein Anker gelöscht wird, werden sämtliche Glieder dieser Kette gelöscht.

4. Jede Kette erhält einen Namen.
   Der Name muß eindeutig sein, weil durch ihn die Kette identifiziert wird.

Kettenordnung

Jede Kette kann sortiert werden, d.h. die Glieder einer Kette werden nach einem Ordnungsbegriff sortiert. Der Anker bleibt in jedem Fall das 1. Glied der Kette.

2.6. Datenbank-Organigramm

Das Organigramm ist die organisatorische Grundlage für die programmiertechnische Formulierung des Strukturteils der Datenbank.

Das Organigramm ist ein Hilfsmittel für Organisatoren und Systemberatung für die problemorientierte Beschreibung der verschiedenen Datenelemente der Datenbank, insbesondere für die Beschreibung der Beziehungen, die zwischen den einzelnen Daten bestehen. Im folgenden werden die Darstellungsformen des Datenbankorganigramms erläutert.

2.6.1. Darstellung der Daten (Satztypen)

Für jeden Satztyp, der in der Datenbank geführt wird, wird ein Rechteck gezeichnet; in dieses Rechteck wird der Satzname eingetragen.

Werden z.B. die bisher konventionell auf Magnetbändern geführten Kundendateien, bestehend aus den Kundenstammsätzen, und die Auftragsdatei, bestehend aus den Kundenaufträgen, in die Datenbankorganisation einbezogen, dann werden die entsprechenden Daten wie in Bild 2.8 dargestellt.

Bisher: Dateien

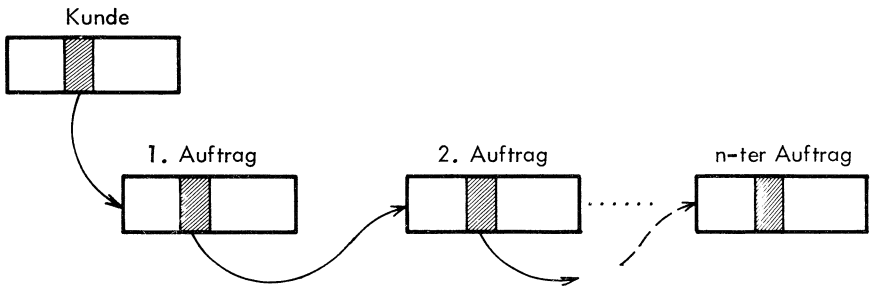
DBS-Organisation: Satztypen

Bild 2.8 Satztypen

2.6.2. Darstellung der Beziehungen

Besteht zwischen zwei verschiedenen Satztypen eine (auswertbare) Beziehung bzw. Abhängigkeit derart, daß ein Datensatz des einen Typs Grundinformationen enthält, die für eine Gruppe von Sätzen des anderen Typs gelten, dann ist dieser Satztyp der Anker der Kette, deren Glieder sind von ihm "abhängige" Sätze.

Netzwerk:



Organigramm:

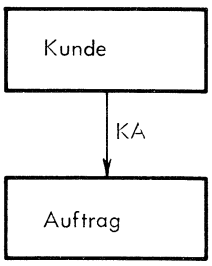


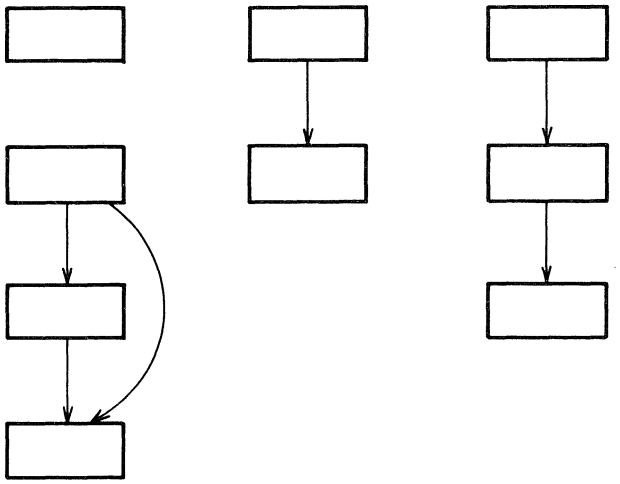
Bild 2.9 Kettenorganigramm und zugehöriges Kettennetzwerk

Diese Beziehung wird folgendermaßen dargestellt:

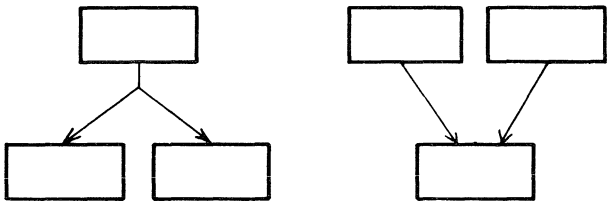
Vom Rechteck, das den Kettenanker repräsentiert, geht ein Pfeil in Richtung des Rechteckes, das die Glieder repräsentiert.

Der Kettenname wird neben dem Pfeil eingetragen (s. Bild 2.9). Die zulässigen Strukturen sind Bild 2.10 zu entnehmen.

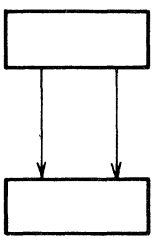
1. Lineare Strukturen:



2. Querbeziehungen:



3. Listenordnung und Ordnung des Verwendungsnachweises



Folgende Ringstruktur ist verboten:

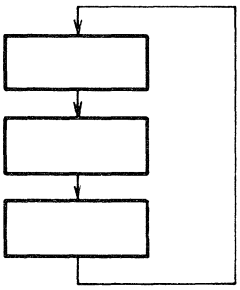


Bild 2.10 Zulässige und verbotene Strukturen

2.6.3. Hierarchien

Für alle Strukturen gilt, daß die Zahl der Stufen (Hierarchien) beliebig ist und daß alle Formen untereinander in einem Organigramm verknüpft werden können.

Am Abschluß der organisatorischen Vorarbeiten für den Aufbau einer Datenbank oder eines Teiles von ihr steht das Datenbankorganigramm.

2.6.4. Beispiel

Die Erzeugnisstrukturdaten einer Baukastenstückliste werden in der Stücklistenkette (STL), aufsteigend sortiert nach Positionsnummern (POS) NAHE dem Kettenanker gespeichert. Kettenanker ist der Teilestammsatz, von dem die Stücklistenkette ausgeht (s. Bild 2.11).

Der Teileverwendungsnachweis (TVN) wird über die TVN-Kette geführt. Anker der TVN-Kette ist der Teilestammsatz des untergeordneten Teiles (sog. Komponente).

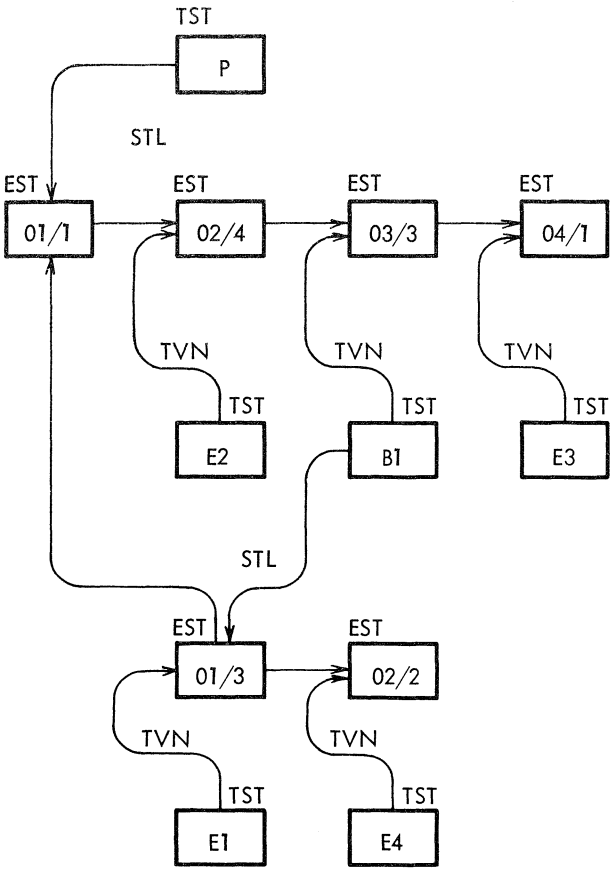
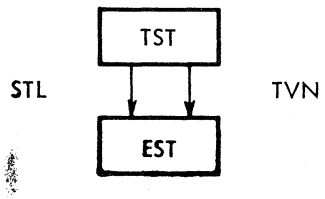


Bild 2.11 Kettennetzwerke der Stückliste aus Bild 2.3

Anstelle des ausführlichen Kettennetzwerkes steht folgendes Organigramm:



2.7. Datenbankbeschreibung und Datenmanipulation

Charakteristisches Merkmal operierender Datenbanksysteme ist ihr Aufbau auf der Grundlage einer problemorientierten Programmiersprache als Basissprache, die dem Benutzer den verfahrensorientierten Rahmen bietet, eine Verarbeitung der Daten im Arbeitsspeicher durchzuführen.

Allen als Basissprache in Betracht kommenden Höheren Programmiersprachen sind zwei Nachteile gemeinsam. Sie besitzen keine leistungsfähigen Statements für die Datenbankbeschreibung und für die Datenbankmanipulation.

Es ist demzufolge naheliegend, die Basissprachen mit spezifischen datenbeschreibenden und datenmanipulativen Statements für ein problemorientiertes, rechnerunabhängiges Arbeiten auf Feld-, Satz- und Dateiebene zu erweitern. Diese Ergänzungsteile zur standardisierten Form der jeweiligen Basissprache sind bei DBS die Datenbankbeschreibungssprache (DBB) und die Datenmanipulationssprache (DMS).

Sämtliche Dienstleistungen, die sowohl Daten vom Arbeitsspeicherbereich in die Datenbank (SPEICHERN) als auch von der Datenbank in den Arbeitsspeicherbereich transferieren (HOLEN), werden als Datentransporte bezeichnet.

Weitere Beispiele solcher Dienstleistungen sind:

Systemdienste:	Datenbankgebiet OEFFNEN, ABSCHLIESSEN
Änderungsdienste:	Feldwerte AENDERN, Datensätze LOESCHEN.

DBS wurde so konzipiert, daß als Basissprache alle gängigen Höheren Programmiersprachen verwendet werden können. Dies sind insbesondere die international anerkannten und genormten Sprachen ALGOL, COBOL und FORTRAN. Der Anschluß an ALGOL und FORTRAN wird zu einem späteren Zeitpunkt realisiert.

Die Beziehungen zwischen den Datenbeschreibungs- und Datenmanipulationssprachen und den Basissprachen zeigt Bild 2.12.

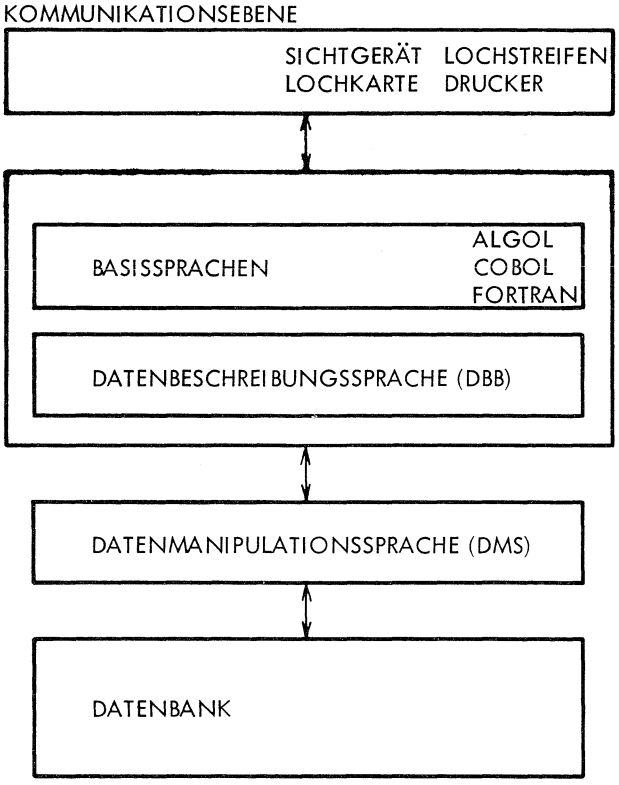


Bild 2.12 Stellung der Sprachteile

3.1. Speicherorganisation

3.1.1. Datenbankträger

Als Datenbankträger werden Direktzugriffsspeicher eingesetzt. Die an den TR 440 angeschlossenen Direktzugriffsspeicher sind:

WSP 414 (Wechselplattenspeicher)
PSP 600 (Festplattenspeicher).

Jedes Datenbankgebiet muß vollständig auf einem Gerätetyp liegen; "gemischte" Speicher als Träger eines Gebietes sind nicht zugelassen. Die verschiedenen Gebiete einer Datenbank dagegen können durchaus auf verschiedenen Gerätetypen liegen.

3.1.2. Elemente der Datenbankspeicherorganisation

Grundlage der Speicherorganisation ist das Gebiets- und das Seitenkonzept.

Jede Datenbank besteht aus mindestens einem Gebiet. Die Zahl der Gebiete einer Datenbank ist beliebig.

Die Elemente der rechnerunabhängigen Speicherorganisation sind:

- Gebiete
- Teilgebiete
- Bereiche
- Seiten

Einen Überblick gibt Bild 3.1. Im folgenden werden diese Elemente ausführlich beschrieben.

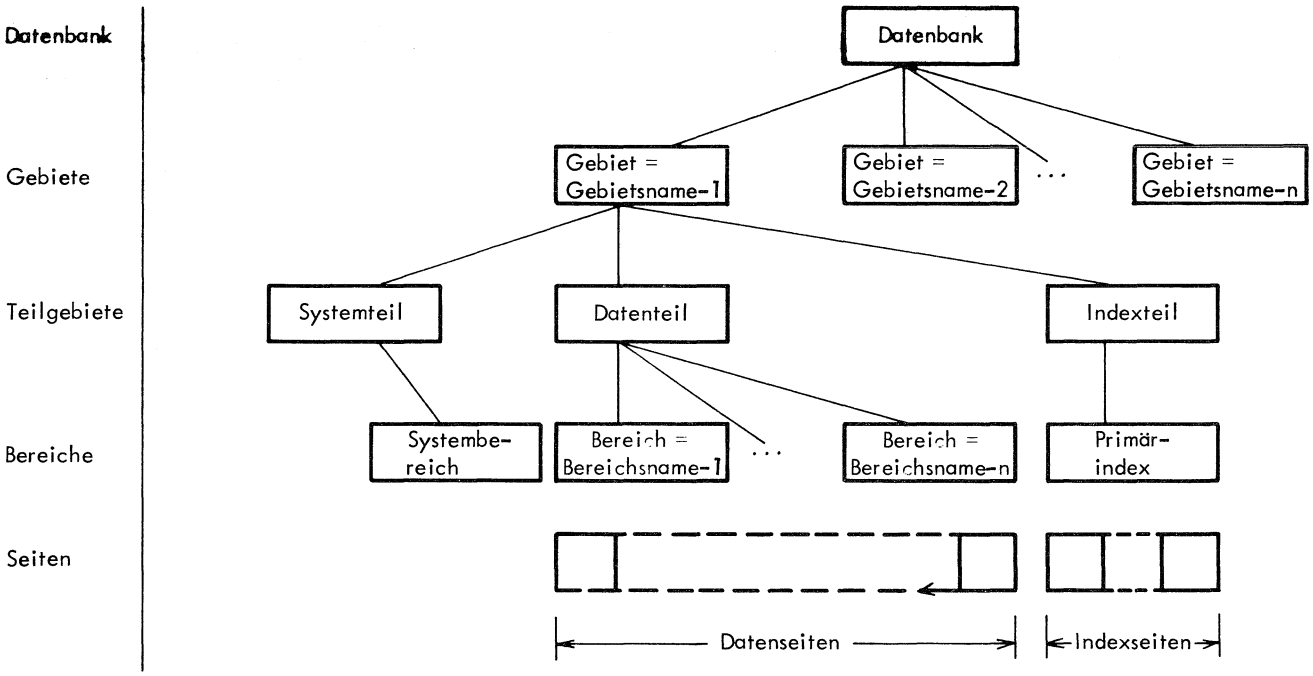


Bild 3.1 Organisatorischer Aufbau der Datenbank

3.1.3. Gebietsorganisation

Systeminterner Aufbau der Speichergebiete

Jede Datenbank besteht aus mindestens einem Gebiet. Jedes Speichergebiet einer Datenbank wird systemintern in mindestens zwei, höchstens aber in drei Teilgebiete gegliedert (s. Bild 3.2).

Teilgebiet 1

Jede Gebietsbeschreibung wird in dem vom DBS-Übersetzer automatisch definierten und angelegten 1. Teilgebiet am Anfang des Speichergebietes als Quellcode und in Form von Steuertabellen für den Datenbankprozessor abgelegt.

Teilgebiet 2

Es wird mindestens ein Datenbereich im Gebietsteil der Datenbankbeschreibung vom DBS-Anwender definiert und vom DBS-Formatierer eingerichtet. Die Datenbereiche dienen der Speicherung der Datenbestände.

Teilgebiet 3

Werden im Speichergebiet Datensätze index-sequentiell gespeichert, dann definiert der DBS-Übersetzer automatisch Indexbereiche, deren Länge und Aufbau (Anzahl der Indexstufen usw.) er aus der Anzahl der index-sequentiell zu speichernden Datensätze und der zugehörigen Schlüssellänge errechnet.

Die Indexbereiche werden vom DBS-Formatierer eingerichtet.

Jedes Speichergebiet einer Datenbank wird also mindestens in zwei, höchstens in drei Teilgebiete unterteilt. Bild 3.3 gibt einen Überblick über die Speichergebiete der Datenbank einer Bibliothek (vgl. auch Bild 2.1, Absch. 2.2).

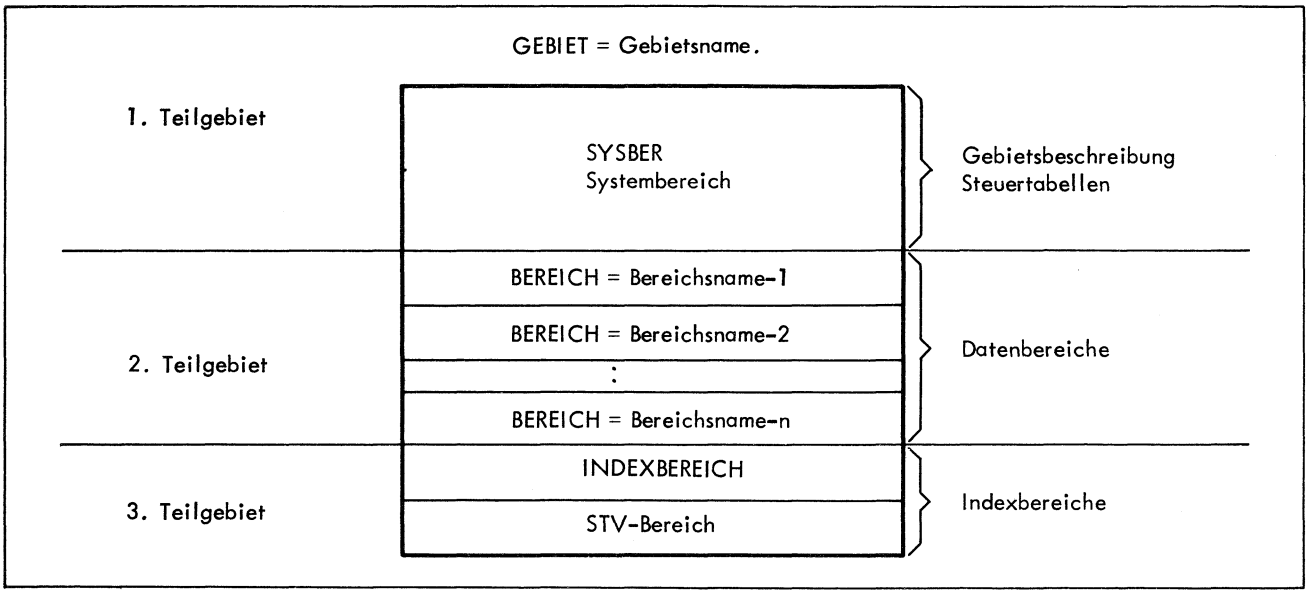


Bild 3.2 Stellung der Teilgebiete und Bereiche in einem Gebiet

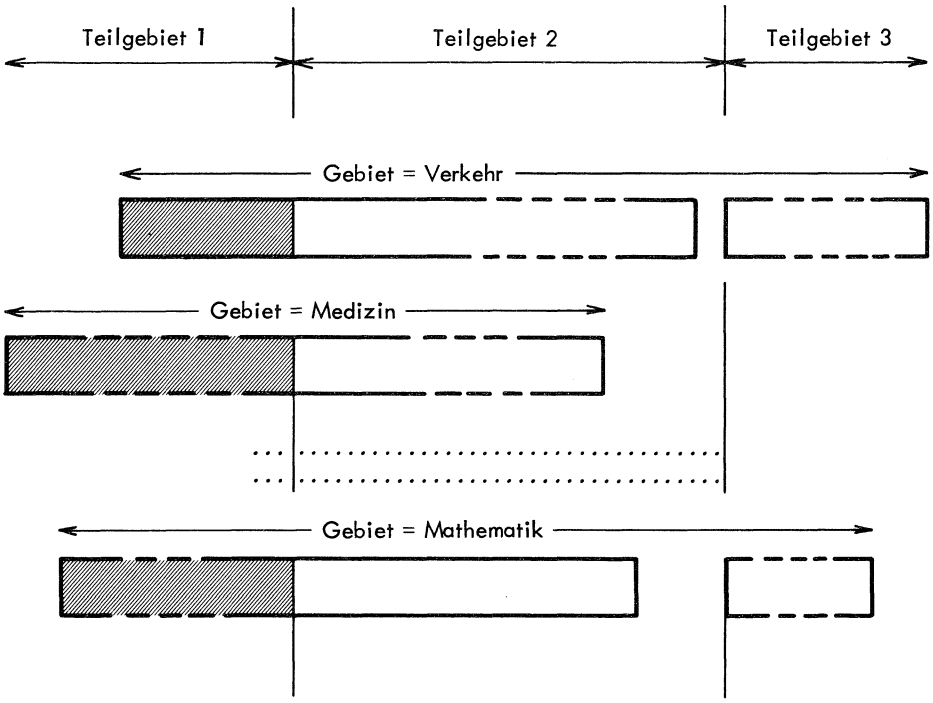


Bild 3.3 Speichergebiete einer Bibliotheks-Datenbank

Gebietslänge

Die Gebietslänge ist die Summe der Längen der Teilgebiete, gemessen in Seiten. Sie wird vom DBS-Übersetzer automatisch errechnet.

Systembereich

Im Systembereich liegen die vom DBS-Übersetzer erstellten Steuertabellen.
Der Systembereich wird auf volle Seitenlänge gerundet.

Länge Teilgebiet 2

Die Länge von Teilgebiet 2 wird vom DBS-Anwender durch den höchsten Wert des Parameters * LAGE = VON SN-1 BIS SN-1 festgelegt.

Teilgebiet 2 muß mindestens eine Seite lang und darf höchstens 2^{18} -2 Seiten lang sein. Die Seitenlänge und die Seitenzahl bestimmen im wesentlichen die Gebietslänge.

Länge Teilgebiet 3

Teilgebiet 3 kann fehlen. Dies ist stets dann der Fall, wenn in das betreffende Gebiet kein Satztyp index-sequentiell gespeichert wird. Die Länge von Teilgebiet 3, gemessen in Seiten, errechnet der DBS-Übersetzer aus der Anzahl der index-sequentiell zu speichernden Datensätze und der Schlüssellängen.

3.1.4. Seitenorganisation

Die Daten- und Indexbereiche eines jeden Datenbankgebietes werden durch den DBS-Formatierer in Seiten gleicher Länge eingeteilt. Die Länge einer Seite ist zweckmäßigerweise ein ganzzahliges Vielfaches einer Sektorlänge des eingesetzten Direktzugriffsspeichers, auf dem das Datenbankgebiet während des Programmlaufes liegt.

Ein Sektor ist die kleinste hardwaremäßig adressierbare Einheit des Direktzugriffsspeichers. Die Sektoren der in TR 440-Installationen eingesetzten Direktzugriffsspeicher haben eine Länge von 128 TR 440 GW, dies sind 768 Bytes.

Die Seitenlänge kann mit dem Parameter SEITENLAENGE= ... vom Anwender vorgeschrieben werden.
Die minimale Seitenlänge ist 768 Bytes (= 128 GW), die maximale Seitenlänge kann 6144 Bytes (= 1024 GW) betragen. Fehlt der Parameter SEITENLAENGE= ... in der Datenbankbeschreibung, dann wird automatisch die Seitenlänge auf 1536 Bytes (= 256 GW) eingestellt.

Die Seiten des Daten- und Indexbereiches werden aufsteigend beginnend mit der Seitennummer 000 001, durchnumeriert. Je Datenbankgebiet sind höchstens $2^{18}-2=262\,142$ Datenseiten zugelassen.

Datenseiten

Jede Datenseite enthält ihren Seitenkennsatz und kann bis zu 64 Datensätze verschiedenen Typs und verschiedenen Formats als sog. Linien aufnehmen. Die Liniennummern (LN) laufen von 00 bis 63 einschließlich (Bild 3.4).

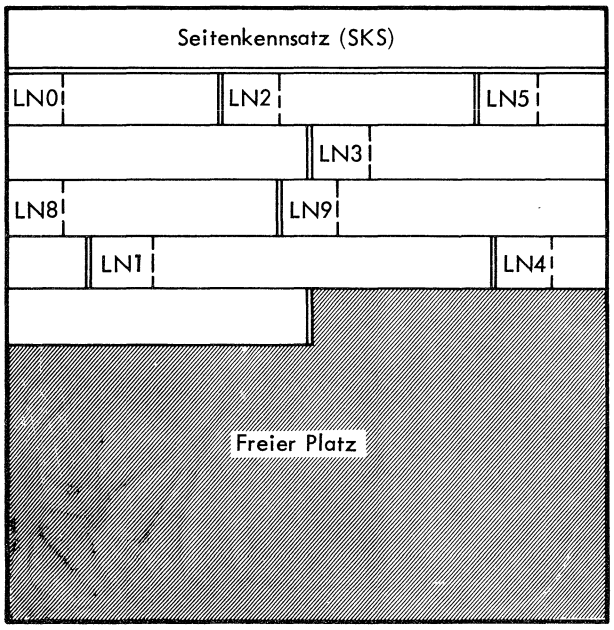


Bild 3.4 Datenseite

Seitenkennsatz

Der Seitenkennsatz (SKS) wird vom DBS-Formatierer aufgebaut und in die ersten drei GW der Seite eingetragen.

Der SKS gibt sowohl Auskunft über die Belegung der Seite als auch über die in ihr z.Z. freie Speicherkapazität.

Auf den Seitenkennsatz folgen unmittelbar die Datensätze und der freie Platz bei nicht vollständiger Seitenbelegung. Bei Einspeicherung eines Satzes in eine Seite wird zunächst geprüft,

- ob noch genügend Platz in der Seite
- und eine freie Liniennummer vorhanden ist.

Werden beide Bedingungen erfüllt, wird der Satz - versehen mit der kleinsten freien Liniennummer - in die Seite eingespeichert. Andernfalls wird die nächste Seite untersucht usw., bis ein freier Platz gefunden ist, auf den dieser Satz gespeichert werden kann.

Wird ein Satz aus einer Seite gelöscht, dann wird diese Seite in der Weise reorganisiert, daß der freie Speicherplatz immer am Ende der Seite liegt. Durch diese Organisation ist das Problem der freien Speicherplatzverwaltung gelöst.

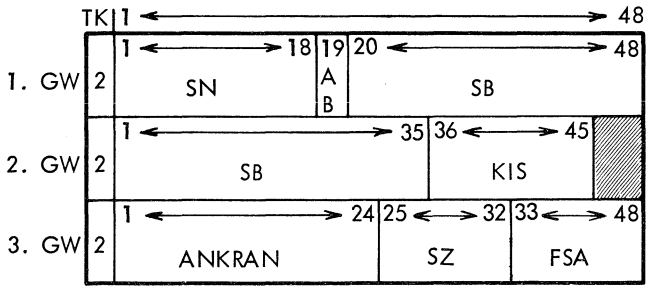


Bild 3.5 Aufbau des SKS für Datenseiten

Der Seitenkennsatz für die Datenseiten hat eine Länge von 3 GW. Die Typenkennung dieser drei GW ist mit TK=2 festgelegt.

Symbole:

Seitennummer - SN:

18 Bit; Bit 1 - 18 / 1. GW
Die im SKS eingetragene Seitennummer ist eine logische Seitennummer; die Adressierung ist gebietsrelativ. Aus der logischen Seitennummer wird vom DBS I/O-Controller die physische Plattenadresse errechnet.
Die Seitennummer ist binär verschlüsselt.
Die logischen Seitennummern beginnen mit 000 001 und enden mit der Seitennummer 262 142.

Abspeicherbit - AB:

1 Bit; Bit 19 / 1. GW
Das Abspeicherbit gibt Auskunft darüber, ob auf eine im zentralen Arbeitsspeicher befindliche Seite schreibend zugegriffen wurde.
(SPEICHERN ..., AENDERN..., LOESCHEN).

AB = 1:

Auf die Seite wurde schreibend zugegriffen. Die Seite muß aufgrund einer durchgeführten Änderung ihres Inhalts in die Datenbank zurückgeschrieben werden.

Das AB-Bit wird beim Zurückschreiben auf die Platte gelöscht.

AB = 0:

Auf die Seite wurde nur lesend zugegriffen. Eine Veränderung des Seiteninhalts wurde nicht vorgenommen. Ein Rücktransport dieser Seite in die Datenbank ist unnötig.

Wird der Arbeitsspeicher, der durch diese Seite belegt ist, angefordert, so kann die Seite überschrieben werden.

Seitenbelegung - SB:

64 Bit; Bit 20 - 48 / 1. GW
Bit 1 - 35 / 2. GW
Die Seitenbelegung gibt Auskunft darüber, welche Liniennummern einer Seite belegt sind und welche Liniennummern noch frei sind.

Die Bits, deren zugehörige Linien noch frei sind, stehen auf 0, die Bits belegter Linien stehen auf 1. Wird ein Datensatz in eine Seite geladen, dann wird ihm die derzeitig kleinste freie Liniennummer zugeteilt. Diese Nummer wird im Leitwort (LTW) des Satzes eingetragen, und das entsprechende Seitenbelegungsbit wird auf 1 gesetzt.

Wird ein Satz aus einer Seite gelöscht, dann wird das Seitenbelegungsbit auf 0 gesetzt, die Linie kann durch einen neu einzuspeichernden Satz wieder belegt werden.

Kapazität in der Seite - KIS:

10 Bit; Bit 36 - 45 / 2. GW
In dieses Feld wird durch den DBS-Formatierer die maximale Speicherkapazität der Seite, gemessen in GW, eingetragen.

Die maximale Speicherkapazität ergibt sich aus der durch den Parameter SEITENLAENGE = ... definierten Seitenlänge minus 3 GW für den Seitenkennsatz.

Das Feld KIS unterliegt einer ständigen Veränderung während eines Programmlaufes.

Wird ein Satz in eine Seite geladen, dann wird KIS um die eingespeicherte Satzlänge reduziert; wird ein Satz aus der Seite gelöscht, dann wird KIS um die gelöschte Satzlänge erhöht.

Ankeradresse der Randomkette - ANKRAN:

24 Bit; Bit 1 - 24 / 3. GW
In das Feld ANKRAN wird die Satzadresse des ersten randomgespeicherten Satzes eingetragen, der logisch zu dieser Seite gehört.
Müßte dieser Satz aufgrund eines Seitenüberlaufs in einer anderen Seite abgelegt werden, dann steht in ANKRAN die Adresse dieses Satzes in der neuen Seite.

Satzzähler - SZ

10 Bit, Bit 24 - 32 / 3. GW
SZ wird vom DBS-Formatierer mit dem Wert der Parameterangabe 'VERTEILUNG = Zahl-1' vorbesetzt.
Wird ein Satz gespeichert, auf den sich der Parameter 'VERTEILUNG' bezieht, dann wird der Inhalt dieses Feldes um den Wert 1 vermindert. Ist der Wert des Feldes SZ = 0, dann kann kein weiterer Satz mehr in diese Seite geladen werden, auf den sich der Verteilungsparameter bezieht.

Freie Satzadresse - FSA:

16 Bit; Bit 25 - 48 / 3. GW
FSA ist vom DBS-Formatierer mit dem Wert drei vorbesetzt. Dieses Feld unterliegt während eines Programmlaufs einer ständigen Veränderung.

Der Inhalt des Feldes FSA zeigt an, ab welchem GW innerhalb der Seite ein neuer Satz geladen werden kann.

Der Wert des Feldes dient gleichzeitig als Abbruchkriterium beim Durchsuchen einer Seite.

Die Felder AB, SB und ANKRAN werden durch den DBS-Formatierer standardmäßig mit Null besetzt.

Indexbereich

Der Indexbereich dient dem schnellen logisch sequentiellen Abspeichern und Wiederauffinden von Informationen.

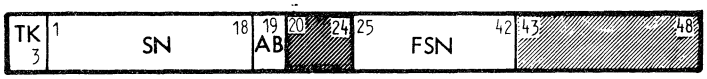
Vom Formatierer wird in Abhängigkeit von Länge der Schlüssel und Anzahl der Datensätze die maximal mögliche Anzahl der Indexstufen ermittelt.
DBS-440 arbeitet mit einem einstufigen (Master- und Stellvertreterliste) bis vierstufigen Index (Master, Index 1, Index 2, Index3, Stellvertreterliste). Beim Neuaufbau der Datenbank wird zunächst mit einem einstufigen Index gearbeitet. Die weiteren Indexstufen werden bei Bedarf automatisch angelegt.
Im Master und im Index werden Indexsätze eingetragen. Jeder Indexsatz repräsentiert eine Seite. Bei zwei Indexstufen gibt es so viele Indexseiten, wie im Master Indexeintragungen möglich sind und so viele Stellvertreterseiten, wie es Indexeintragungen in allen Seiten der Indexstufen gibt.

Der Indexbereich ist in Seiten gleicher Länge eingestellt. Die Länge der Seiten entspricht der Länge der Datenseiten. Die Seiten werden fortlaufend numeriert, beginnend mit Seitennummer 1 bis Seitennummer 262142 ($2^{18}-2$).
Die Seitennummer 1 wird durch den Master belegt.

Seitenaufbau

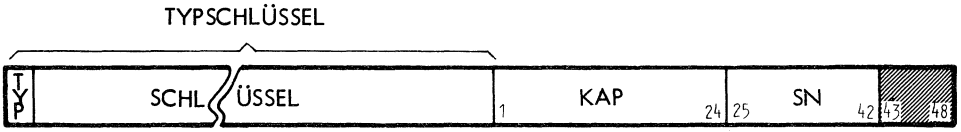
Die Seiten des Indexbereiches haben einen einheitlichen Aufbau.
Das 1. GW der Seite wird durch den Indexseitenkennsatz belegt. Die restliche Kapazität der Seite dient der Aufnahme der Index- bzw. Stellvertretersätze.

Aufbau Indexseitenkennsatz



SN = Seitennummer der Index- bzw. Stellvertreterseite
AB = Abspeicherbit (siehe Seitenkennsatz f. Datenseiten)
FSN = Freie Seitennummer enthält die nächste freie Seitennummer in der jeweiligen Indexstufe.

Aufbau Indexsatz



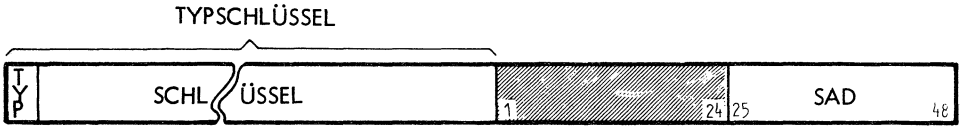
TYP = enthält den Typ des Satzes, zu dem der Schlüssel gehört (1 Byte).

SCHLUESSEL = 1 - 29 Byte
Im SCHLUESSEL ist der höchste Ordnungsbegriff der Seite eingetragen, der im Feld SN steht.

KAP = 24 Bit, Bit 1 - 24
enthält die Anzahl freier Plätze der in SN angegebenen Index- bzw. STV - Seite.

SN = 18 Bit, Bit 25 - 42
enthält die Seitennummer für die die Schlüsseleintragung gilt.

Aufbau Stellvertretersatz



TYP = enthält den Typ des Satzes, zu dem der Schlüssel gehört (1 Byte).

SCHLUESSEL = 1 - 29 Byte
In SCHLUESSEL wird der Ordnungsbegriff des Datensatzes eingetragen.

SAD = 1 HW
enthält die Satzadresse des zugehörigen Datensatzes.

Datenseitentransport

Für den Datentransport zwischen Datenbankträger und Kernspeicher wird im Kernspeicher eine zentrale Arbeitszone, die DBS-Pufferzone, angelegt. Diese Pufferzone ist in Seiten gleicher Länge eingeteilt, die Länge der Pufferseiten entspricht der Länge der Datenseiten. Die Zahl der Pufferseiten, kann der DBS-Anwender für den jeweiligen Operatorlauf in Feld PUFFERZAHL des NVB'S angeben (2 ≦ PUFFERZAHL ≦ 100; Voreinstellung = 4).

Der Datentransport zwischen dem Datenbankträger und den DBS-Puffern erfolgt immer seitenweise. Da innerhalb einer Seite mehrere Datensätze geführt werden, wird mit einem einzigen Datentransport ein Zugriff auf mehrere Datensätze erzielt. Der Vorteil dieses Konzepts wirkt sich dann voll aus, wenn Sätze, die gemeinsam bzw. nacheinander verarbeitet werden sollen, sich in derselben Seite befinden.

Der Datentransport zwischen Kernspeicher und Datenbankspeicher wird normalerweise durch einen der Transportbefehle HOLEN ... bzw. SPEICHERN eingeleitet, wobei sich diese Befehle grundsätzlich auf jeweils einen Datensatz beziehen, systemintern dagegen nur Seiten transportiert werden.

Der DBS-Anwender arbeitet bezüglich der Eingabe/Ausgabe auf der sogenannten logischen Stufe, der DBS Input/Output-Controller dagegen arbeitet auf der sogenannten physischen Stufe (vgl. Bild 3.6).

Der Datenaustausch zwischen DBS-Puffern und dem Arbeitsspeicher des DBS-Anwendungsprogramms erfolgt automatisch.

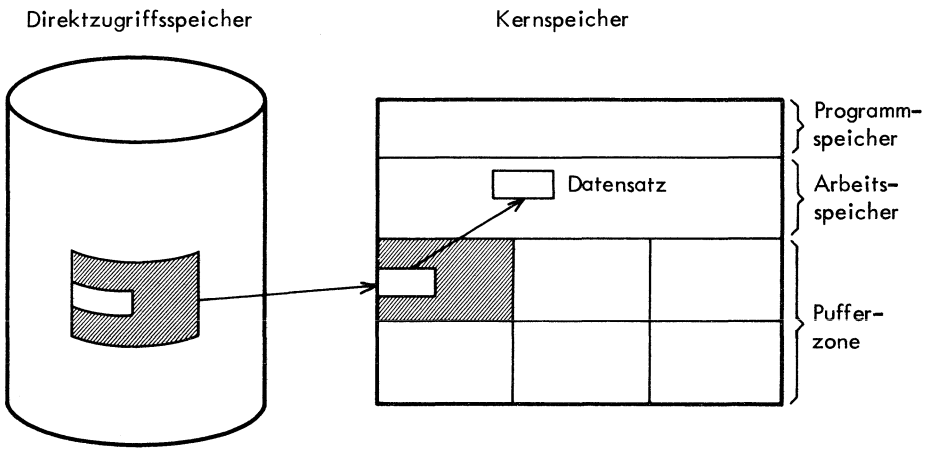


Bild 3.6 Logische und physische Eingabe/Ausgabe

Puffersteuerung

Zur DBS-Pufferzone wird eine Puffergewichtstabelle angelegt, die der Steuerung der DBS-Pufferseiten dient. Für jede reservierte Pufferseite wird 1 GW in der Puffergewichtstabelle angelegt. Die Position des GW innerhalb der Puffergewichtstabelle entspricht der Position der Pufferseite innerhalb der DBS-Pufferzone, so daß eine eindeutige Zuordnung zwischen der Pufferseite und dem GW in der Puffergewichtstabelle besteht.

In das reservierte GW für die Puffergewichtstabelle wird im linken Teil ein Puffergewicht und im rechten Teil die Kernspeicheradresse der zugehörigen Pufferseite eingetragen (s. Bild 3.7).

Das Puffergewicht gibt Auskunft über die Aktivität der Seite in der DBS-Pufferzone.

	TK	Puffergewicht	Pufferadresse
	2	1	Puffer 1
	2	2	Puffer 2
	2	3	Puffer 3
	2	n	Puffer n
Ende-kennung →	3		

Bild 3.7 Puffergewichtstabelle

Die Seite mit dem niedrigsten Puffergewicht ist die aktive Pufferseite, die Seite mit dem höchsten Puffergewicht ist die Seite, auf die am längsten nicht mehr zugegriffen wurde. Bei jedem Datenseitentransport wird zuerst die Pufferzone daraufhin untersucht, ob sich diese angeforderte Seite nicht schon im Puffer befindet. Muß die neue Seite zugeladen werden, so wird sie in die Pufferzone transportiert, wobei die am längsten nicht mehr bewegte Seite in der Pufferzone entweder überlesen werden kann (bei AB = 0) oder zuvor auf dem Datenbankträger zurückgespeichert werden muß, weil das Abspeicherbit AB im Seitenkennsatz auf 1 gesetzt ist.

3.1.5. Datensätze

Bis zu 127 verschiedene Satztypen können für ein Datenbankgebiet definiert werden.

Das Format eines Satzes kann dabei

- fest (ohne Speicheroptimierung)
- variabel (mit Speicheroptimierung)

sein.

Satzaufbau

Jeder Satz besteht immer aus zwei Teilen:

- Steuerteil
- Datenteil

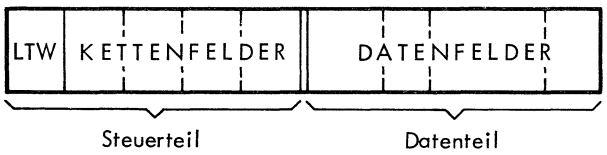


Bild 3.8 Aufbau des Datensatzes

Der Steuerteil, der vom System automatisch aufgebaut und ausgewertet wird, beginnt stets mit einem Leitwort (LTW), dem bis zu 63 Kettenfelder für die Aufnahme von logischen Satzadressen folgen können (s. Bild 3.8).

Der DBS-Anwender hat keinen Zugriff auf den Steuerteil eines Datensatzes.

Leitwort

Das Leitwort wird beim Speichern eines Satzes aufgebaut und dient der Steuerung des Satzes.

Das Leitwort hat eine Länge von 24 Bit = 1 TR 440 HW (s. Bild 3.9).

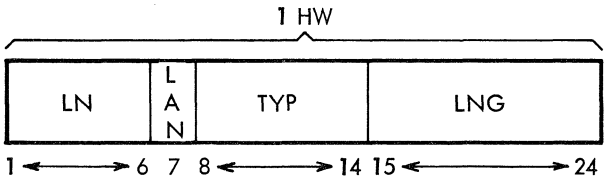


Bild 3.9 Aufbau des Leitwortes

Dabei gilt:

LN = Liniennummer
Sie dient der Adressierung des Satzes innerhalb einer Seite.

LAN = Löschanzeiger
Er gibt Auskunft darüber, ob der Datenteil dieses Satzes noch verarbeitet werden kann (LAN = 0) oder aber bereits logisch gelöscht ist (LAN = 1).

TYP = Bezeichnet den Typ des jeweiligen Satzes.
Den Typ definiert der DBS-Anwender durch den Datenparameter SATZTYP = ...

LNG = Länge des Satzes.
Die Länge des Satzes ergibt sich aus der Länge des Steuerteils plus der Länge des Datenteils.
Die Gesamtlänge des Satzes darf die Seitenlänge minus 3 GW für den SKS nicht überschreiten.

Dabei gilt:

SN = Seitennummer
Mit den für die Seitennummer reservierten 18 Bits können $2^{18}-2$ Seiten (=262142 Seiten) adressiert werden.

LN = Liniennummer
Mit den 6 Bits für die Liniennummern können 64 Sätze pro Seite adressiert werden.

Die Satzadresse ist eine logische Adresse. Mit Hilfe eines Umrechnungsverfahrens wird aus der logischen Satzadresse eine physikalische Plattenadresse errechnet.

3.1.6. Indexsätze

Der Aufbau des Index- bzw. Stellvertretersatzes ist fest. Jeder Satz besteht aus einem eins- bis fünf GW langen Schlüsselfeld und einem 1 GW langen Steuerfeld. Die Gesamtlänge der Index- bzw. Stellvertretersätze ergibt sich aus Länge Schlüsselfeld plus 1 GW für das Steuerfeld. Die Stellvertretersätze unterscheiden sich von den Indexsätzen nur durch den unterschiedlichen Aufbau des Steuerfeldes.

3.1.7. Adressen

Die Satzadresse ist das Identifizierungsmittel, mit dessen Hilfe vom Datenbankprozessor ein Datensatz in der Datenbank eingespeichert und in ihr auch wieder aufgefunden werden kann.

Eine einmal für einen Satz vergebene Satzadresse hat für den Satz solange Gültigkeit, wie der Satz in der Datenbank geführt wird.

Die Satzadresse wird intern auf eine 24 Bit-Adresse abgebildet, wobei die ersten 18 Bit die Seitennummer, die restlichen 6 Bit die Liniennummer innerhalb der Seite angeben (s. Bild 3.10).

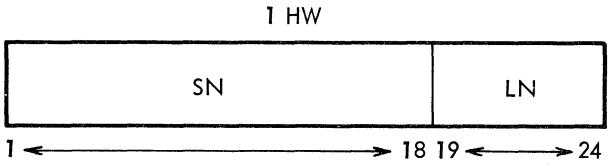


Bild 3.10 Aufbau der Satzadresse

3.1.8. Kettenfelder

Die Anzahl der Kettenfelder eines Satztyps ist abhängig von der Stellung dieses Satztyps in der Datenbankstruktur.

Ein Satztyp muß nicht unbedingt Element einer Kettenstruktur sein, er kann auch völlig "isoliert" existieren. DBS läßt je Satztyp bis zu 63 Kettenfelder zu; eine Verkettung in diesem Umfang dürfte praktisch nicht vorkommen.

Jedes Kettenfeld ist 24 Bit lang. In einem Kettenfeld steht die Satzadresse des logischen Nachfolgers bzw. Vorgängers eines Kettenelements bzw. die Adresse des Kettenankers (s. Bild 3.11).

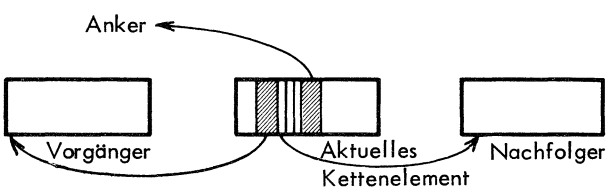


Bild 3.11 Verkettung

3.1.9. Datenteil

Der Datenteil des Satzes ist eine Zusammenfassung von Datenfeldern, der die jeweils zu speichernde Information enthält.
Aufbau, Inhalt und Länge des Datenteils werden in der Datenbeschreibung festgelegt.

Die Länge des Datenteils ist normalerweise von Satztyp zu Satztyp verschieden, jedoch innerhalb eines Satztyps konstant, falls nicht durch den Kompressor eine Datenverdichtung durchgeführt wird.

3.2. Speicherungs- und Verarbeitungsformen

Für die Datenorganisation und für die Verarbeitung der Daten mit DBS stehen dem Anwender fünf Speicherungs- und Verarbeitungsformen zur Verfügung.

- RANDOM
- DIREKT
- NAHE
- SEQUENTIELL
- INDEX-SEQUENTIELL

Die verschiedenen Speicherungsformen erlauben es - zusammen mit der Verkettungstechnik - Daten entsprechend der späteren Verarbeitung zu speichern und somit günstige Durchlaufzeiten zu erzielen.

Die Speicherungsformen werden definiert durch den Parameter ABLAGE = ..., der im Falle der Speicherungsform RANDOM bzw. INDEX-SEQUENTIELL ergänzt wird durch den Parameter SCHLUESSEL = ...

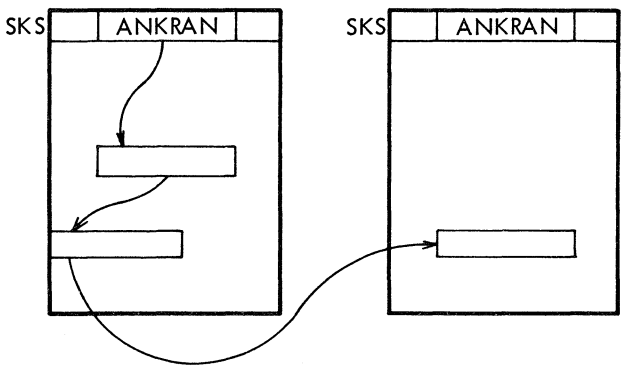


Bild 3.12 Randomkette

3.2.1. Random (RAN)

Aus dem für den Satztyp vorgegebenen Ordnungsbegriff SCHLUESSEL errechnet der Randomgenerator die Seitenadresse, der der Satz aufgrund seines Ordnungsbegriffes angehört. Alle RANDOM zu speichernden Sätze, die logisch zu einer Seite gehören, werden miteinander verkettet. Die Kette wird als Randomkette bezeichnet und wird bei der Speicherungsform RANDOM automatisch vom System angelegt, sie wird also nicht im Strukturteil der Datenbankbeschreibung definiert.

Anker der Randomkette ist der Seitenkennsatz der Seite, dem auch die Sätze mit derselben Randomnummer logisch zugeordnet sind. Die Adresse des ersten Gliedes der Randomkette wird in das Feld ANKRAN des Seitenkennsatzes eingetragen.

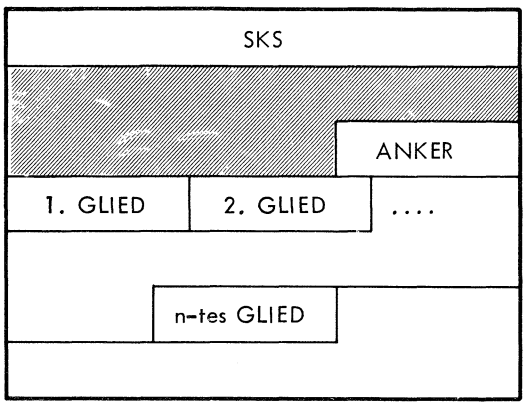
Kann ein Satz wegen Seitenüberlaufs nicht in die vom Randomgenerator errechnete Seite gespeichert werden, dann wird dieser Satz physikalisch in die nächste Seite abgelegt, die über genügend Speicherkapazität verfügt. Dieser Überlaufsatz wird aber in die Randomkette der Seite eingefügt, zu der er logisch gehört (s. Bild 3.12).

In der Randomkette können Sätze unterschiedlichen Typs geführt werden. Die Randomkette ist aufsteigend sortiert nach Typ und innerhalb des Typs nach Schlüssel. Die Randomspeicherung läßt keine Duplikate desselben Satztyps zu.

3.2.2. DIREKT (DIR)

Charakteristische Eigenschaft der Speicherungsform DIREKT ist, daß der DIREKT zu speichernde Satz keinen identifizierenden Feldnamen enthält.

Dem Benutzer wird nach dem Speicherungsvorgang im Feld DIREKTADRESSE im Nachrichtenvermittlungsblock (NVB s. 4.4.1) die Adresse des gerade gespeicherten Satzes bereitgestellt. Es ist Aufgabe des DBS-Anwenders selbst eine Zuordnung Ordnungskriterium - Satzadresse in Form einer externen Tabelle zu führen, wenn er diese Sätze auch DIREKT verarbeiten möchte.



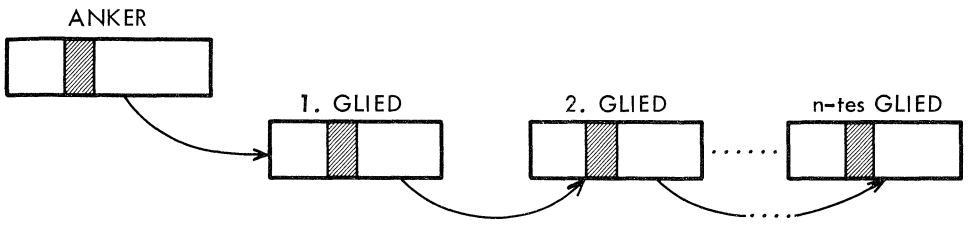


Bild 3.13 Speicherungsform NAHE

3.2.3. NAHE

Mit der Speicherungsform NAHE ist es möglich, die Gliedsätze einer Kette physikalisch möglichst nahe dem Ankersatz zu speichern. Ob ein Satz jedoch auch in dieselbe Seite geladen werden kann, in der sich der Anker befindet, ist abhändig von der freien Speicherkapazität der Seite.

Der Vorteil dieser Speicherungsform liegt darin, daß mit dem Wiederauffinden des Ankersatzes die zugehörigen Gliedsätze zur Verfügung stehen, sofern sie auch physikalisch in der gleichen Seite wie der Anker liegen. Bei der Bearbeitung der Gliedsätze müssen dann keine zusätzlichen Plattenzugriffe durchgeführt werden. Um diesen Vorteil bei der Verarbeitung voll auszunutzen, sollten die Gliedsätze unmittelbar nach dem Anker eingespeichert werden (s. Bild 3.13).

3.2.4. SEQUENTIELL (SEQ)

Die sequentiell zu speichernden Sätze werden (nach Satztyp) in einen nur für diesen Satztyp reservierten Bereich innerhalb der gesamten Datenbank zusammengefaßt. Derartige Bereiche werden auch als SEQ-Dateien bezeichnet (s. Bild 3.14).

INTEGRIERTER DATENBEREICH	SEQ-DATEI A	SEQ-DATEI B

Bild 3.14 SEQ-Dateien

Die SEQ-Dateiorganisation entspricht der Magnetbandorganisation, d.h. die Sätze werden physikalisch hintereinander abgelegt, ohne daß Rücksicht auf einen Ordnungsbegriff oder eine bestimmte Sortierfolge genommen werden muß.

Sequentiell gespeicherte Daten können entweder durch Absuchen der entsprechenden Datei oder bei der Bearbeitung als Gliedsätze einer Kette wiederaufgefunden werden.

3.2.5. INDEX-SEQUENTIELL (ISQ)

Die index-sequentiell zu speichernden Datensätze werden mit Hilfe ihres Ordnungsbegriffes (SCHLUESSEL) eingespeichert und können mittels dieses Ordnungsbegriffes wiederaufgefunden werden. Die index-sequentielle Speicherung ermöglicht neben der gezielten Verarbeitung eines bestimmten Datensatzes auch eine Verarbeitung der Daten in aufsteigender und absteigender Sortierfolge. Die Daten müssen nicht vorsortiert werden; ein Vorsortieren der Daten beschleunigt jedoch das Speichern.

Aufgrund des Parameters ABLAGE = INDEX-SEQUENTIELL wird vom DBS-Übersetzer automatisch ein Indexbereich angelegt. Dieser Parameter kann ergänzt werden durch den Parameter VERTEILUNG (siehe 4.16).

Der Indexbereich wird standardmäßig hinter dem Teilgebiet 2 des Speichergebietes angelegt.

Indexbereich

Der Indexbereich nimmt den Master die Indexseiten und die Stellvertreterlisten auf.
Der Indexbereich ist in Seiten gleicher Länge eingeteilt. Die Seitenlänge entspricht der definierten Seitenlänge des Datenbereiches.

Der gesamte Indexbereich kann auch als Inhaltsverzeichnis angesehen werden, mit dessen Hilfe das Wiederauffinden gespeicherter Daten beschleunigt wird.

DBS arbeitet mit einem ein- bis vierstufigen Index. Die Zahl der Indexstufen errechnet der DBS-Übersetzer automatisch aus der Anzahl der index-sequentiell zu speichernden Daten (INHALT = ...), der Länge des Stellvertretersatzes und der definierten Seitenlänge der Datenbank.

Indexstufen

Für den gesamten Indexbereich wird ein Masterindex (Hauptindex) angelegt. Der Materindex entspricht der 1. Indexstufe und ist immer vorhanden.

Der Index 1 (2. Indexstufe), Index 2 (3. Indexstufe) und Index 3 (4. Indexstufe) wird bei Bedarf vom DBS-Übersetzer automatisch angelegt.

Stellvertreter (STV)

Der DBS-Befehl SPEICHERN .. erstellt für einen index-sequentiell zu speichernden Datensatz automatisch einen Stellvertreter.
Der Stellvertretersatz nimmt den Satztyp, den Ordnungsbegriff und die logische Adresse des zugehörigen Datensatzes auf. Satztyp und Ordnungsbegriff des Datensatzes ergeben zusammen den Ordnungsbegriff des Stellvertretersatzes.

Die Stellvertretersätze werden in einer logisch aufsteigenden Reihenfolge in die Stellvertreterliste eingeordnet. Die zugehörigen Datensätze werden im Datenbereich abgelegt.

Neu hinzukommende Stellvertretersätze werden, wenn sie logisch zu einer bereits angelegten Stellvertreterseite gehören, in diese eingefügt. Kann ein Stellvertretersatz aus Kapazitätsgründen nicht mehr in die Seite eingeordnet werden, zu der er logisch gehört, dann wird diese Seite geteilt und der Stellvertretersatz dann eingeordnet.

3.3. Die Datenintegration

3.3.1. Kettentechniken

Kettendefinition

Mit Hilfe direkt adressierbarer Massenspeicher kann man Informationen verhältnismäßig schnell speichern und wiederauffinden, aber eine integrierte Datenhaltung ist damit noch nicht erreicht. Dazu bedarf es einer besonderen Strukturierungstechnik, einer weiteren Datenstufe: der Kette. Mit ihrer Hilfe ist es möglich, jede funktionelle oder organisatorische Abhängigkeit von Datenbeständen in einem Unternehmen, in einer Verwaltung sichtbar zu machen mit dem Ergebnis eines einzigen integrierten Datenbestandes, der nach beliebigen Gesichtspunkten verarbeitet werden kann.

Eine Kette ist eine Folge von logisch zusammengehörigen Sätzen. Sie besteht aus einem einzigen Anker und keinem oder beliebig vielen Gliedern. Der Ankersatz erfüllt in einer Kette zwei Funktionen, einmal ist er der Einsprungspunkt in eine Kette und zum anderen enthält er Informationen, die für die gesamte Kette gelten. Die ergänzenden, variablen Informationen stehen in den Gliedsätzen.

In DBS sind die Ketten nicht geschlossen, d.h. es gibt keinen Verweis von dem letzten Gliedsatz einer Kette auf den Ankersatz; dafür enthält der letzte Gliedsatz ein Kettenendezeichen (s. Bild 3.15).

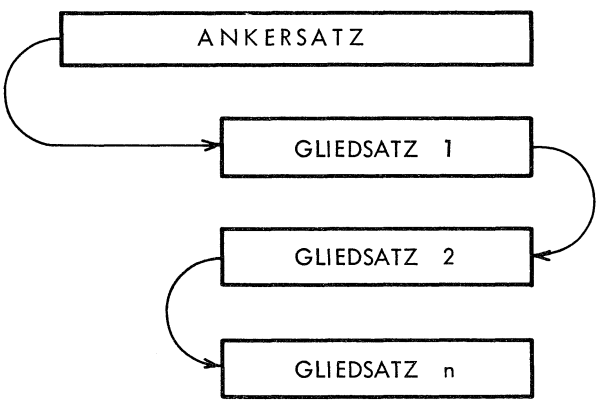


Bild 3.15 Offene Kette mit Ankersatz und Gliedsätzen

Sinn einer Kette ist es, logisch zusammengehörige Daten miteinander zu verbinden. So kann z.B. ein bestimmter Vergasertyp in verschiedenen Automodellen Verwendung finden. In der Stückliste jedes dieser Modelle taucht somit derselbe Vergasertyp auf, bei der Einspeicherung der Stückliste jedoch erscheint der Vergasertyp physikalisch gesehen nur einmal, denn DBS baut mit dem Vergasertyp als Ankersatz eine Teileverwendungskette auf, deren Gliedsätze darüber Auskunft geben, wo und mit welcher Stückzahl der Vergasertyp überall vorkommt. Der Vorteil von DBS zeigt sich hier in reduzierter Redundanz und folglich auch in einem entsprechend geringeren Änderungsaufwand.

Jeder denkbare Kettenaufbau ist erlaubt, mit der einzigen Einschränkung, daß ein Satz nicht Glied von sich selbst sein kann, weder direkt noch über mehrere Ketten (vgl. verbotene Ringstruktur im Datenbankorganigramm, s. 2.6.2).

Die Kettenfelder

Dem Datenteil eines Satzes gehen, falls der Satz wenigstens einer Kette angehört, Kettenfelder voraus. Mit Hilfe dieser Kettenfelder wird die Verbindung zwischen dem Ankersatz und seinen Gliedsätzen einerseits und zwischen den Gliedsätzen andererseits hergestellt. Ein Kettenfeld enthält die logische Adresse des nächsten Satzes, und zwar des nächsten in der logischen Folge einer Kette.

Der Ankersatz enthält obligatorisch ein Nachfolgerfeld, das für die Aufnahme der logischen Adresse des ersten Gliedsatzes einer Kette bestimmt ist. Im Gliedsatz können bis zu drei Kettenfelder pro Kette angelegt werden. Auch hier ist für die normale Verarbeitung vom Ankersatz bis zum letzten Gliedsatz einer Kette ein Nachfolgerfeld obligatorisch. Soll die Kette auch in umgekehrter Richtung verarbeitet werden können, also vorgängerverarbeitbar sein, dann muß in jedem Gliedsatz zusätzlich zum Nachfolgerfeld ein Vorgängerfeld definiert werden, das die logische Adresse des jeweiligen Vorgängersatzes aufnimmt (s. Bild 3.16).

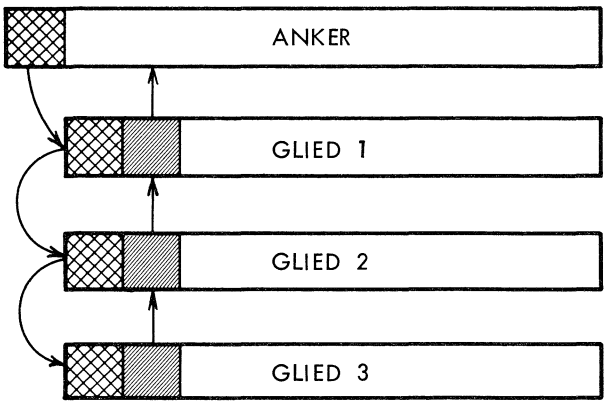


Bild 3.16 Vorgängerverarbeitbare Kette

Darüber hinaus kann der Benutzer vereinbaren, daß eine Kette ankerverarbeitbar sein soll, d.h. jeder Gliedsatz soll eine Verweisadresse auf den Ankersatz der Kette enthalten (s. Bild 3.17).

Die Verkettung zum Ankersatz bietet die Möglichkeit, von jedem Gliedsatz direkt den Ankersatz zu finden. Dies wird besonders bei der Stücklistenorganisation benötigt.

Besteht eine Kette nur aus einem Anker, so enthält das Nachfolgerfeld ein Kettenendezeichen. Dasselbe gilt für das Nachfolgerfeld des jeweils letzten Gliedsatzes einer Kette.

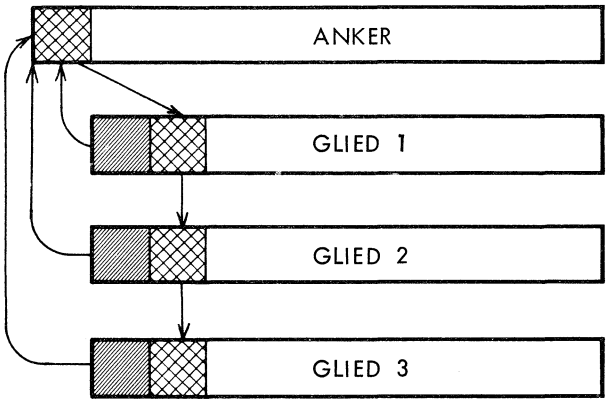


Bild 3.17 Ankerverarbeitbare Kette

3.3.2. Die Kettenordnungen

Unter Kettenordnung ist die logische Einordnung von Gliedern in eine Kette zu verstehen. DBS unterscheidet zwischen sortierter und unsortierter Kettenordnung. Die für eine Kette vorgesehene Kettenordnung wird mit dem Parameter "EINORDNUNG = ..." definiert.

Sortierte Kette

Bei der sortierten Kette werden die Glieder nach einem Ordnungsbegriff auf- oder absteigend sortiert. Der Ordnungsbegriff muß Bestandteil sämtlicher Glieder sein. Bei der auf- bzw. absteigenden Sortierung können als Glieder einer Kette nur Sätze gleichen Typs geführt werden (vgl. Bild 3.18 und Bild 3.19).

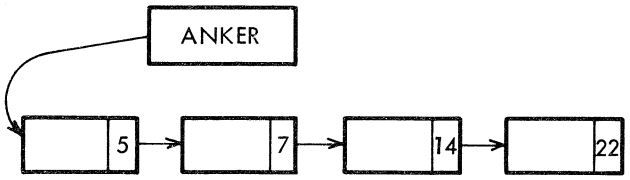


Bild 3.18 Kette mit EINORDNUNG = SORTIERT AUFSTEIGEND NACH Feldname FELD

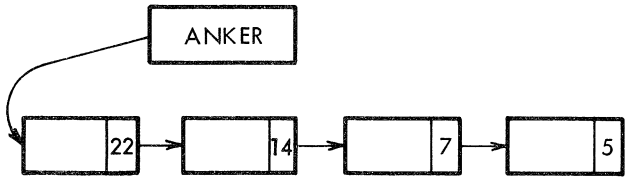


Bild 3.19 Kette mit EINORDNUNG = SORTIERT ABSTEIGEND NACH Feldname FELD

Für die auf- bzw. absteigend sortierte Kette können Duplikate verboten bzw. erlaubt sein. Als DBS-Standard sind Duplikate verboten. Sollen Duplikate in der sortierten Kette geführt werden, dann muß dies mit dem Parameter "DUPLIKATE = ERLAUBT" angegeben werden.

Für die Kette, in der Glieder unterschiedlichen Typs geführt werden, besteht die Möglichkeit der Sortierung nach Typ. Die Sortierung ist grundsätzlich aufsteigend. Bei der Sortierung nach Typ kann kein weiterer Sortierbegriff angegeben werden (s. Bild 3.20).

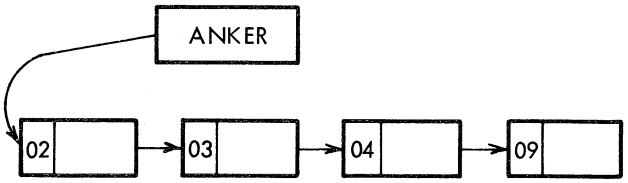


Bild 3.20 Kette mit EINORDNUNG = SORTIERT NACH TYP

Unsortierte Kette

In unsortierte Ketten werden die Kettenglieder ohne Rücksicht auf Sortierbegriff oder bereits bestehende Sortierfolgen in die Kette eingeordnet. Für die unsortierte Kette bestehen vier Möglichkeiten der Einordnung (s. Bild 3.21).

EINORDNUNG = AM KETTENANFANG

"Einordnung am Kettenanfang" besagt, daß ein neu hinzukommendes Glied immer erstes Glied der Kette wird.

EINORDNUNG = AM KETTENENDE

Neu hinzukommende Glieder werden an das Kettenende angefügt. Im Anker der Kette wird automatisch ein Kettenfeld angelegt, in das die Adresse des letzten Gliedes der Kette eingetragen wird.

EINORDNUNG = DAVOR

Ein neu hinzukommendes Glied wird logisch vor das zuletzt bearbeitete Glied der Kette (aktuelles Kettenelement) eingefügt. Wird das erste Glied der Kette geladen, dann wird es logisch hinter den Anker angefügt. Bei der Einordnung DAVOR wird automatisch vom Datenbankprozessor die Vorgängerverkettung durchgeführt. Wird zusätzlich die Verkettung mit dem Vorgänger angegeben, dann wird trotzdem nur ein Kettenfeld für die Vorgängerverkettung angelegt.

EINORDNUNG = DANACH

Ein neu hinzukommendes Glied wird logisch hinter das zuletzt bearbeitete Glied der Kette (aktuelles Kettenelement) eingefügt.

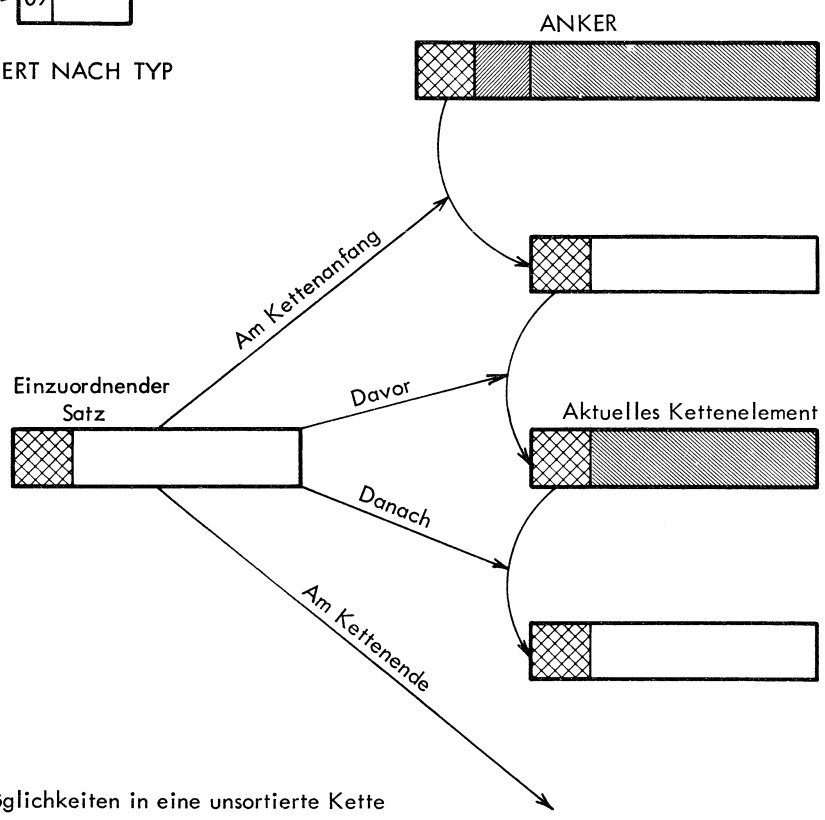


Bild 3.21 Einordnungsmöglichkeiten in eine unsortierte Kette

Der Vorteil der Einordnung in eine nicht sortierte Kette liegt in einer schnellen Speicherung von Gliedsätzen, da jede Art von Sortierung sehr zeitaufwendig ist. Daraus folgt aber nicht, daß in unsortierten Ketten eine sortierte Speicherung nicht möglich ist. Gibt man beispielsweise bei EINORDNUNG AM KETTENENDE oder DANACH die Datensätze aufsteigend sortiert ein, so ist die Kette automatisch aufsteigend sortiert.

In jedem der in Bild 3.22 abgebildeten Sätze steht dessen logische Satzadresse. Der Einsprungspunkt in die Kette sei der Ankersatz und die Kette nur nachfolgerverarbeitbar. Wie sieht die Kettentabelle aus, wenn zuerst der Ankersatz, dann der erste Gliedsatz usw. gefunden worden ist? Vergleiche folgendes Bild 3.23.

Der Einsprungspunkt in die Kette sei jetzt Glied 3 und die Kette selbst nur nachfolgerverarbeitbar. Da die Kette weder vorgänger- noch ankerverarbeitbar ist, kann weder der Anker noch Glied 2 noch Glied 1 gefunden werden. Nach dem Zugriff auf Gliedsatz 3 sieht die Kettentabelle wie in Bild 3.24 aus.

3.3.3. Kettentabellen

Kettentabellen werden vom Vorübersetzer angelegt, von den DBS-Befehlen benutzt und nach Ablauf eines jeden Befehls auf den neuesten Stand gebracht. Für jede Kette wird eine Kettentabelle aufgebaut, auf die zwar der Benutzer nicht zugreifen kann, um deren Bedeutung er jedoch bei der Planung leistungsfähiger Ketten, bei der ANKERWAHL und beim Absetzen von DBS-Befehlen wissen muß.

In einer Kettentabelle ist Platz für vier Eintragungen: für die logische Adresse des Ankersatzes (ANK), des zuletzt bearbeiteten Satzes der Kette (AKT), des logischen Vorgängers (VOR) und des logischen Nachfolgers zum aktuellen Satz (NACH). Beim Arbeiten in einer Kette werden diese stets fortgeschrieben.

ANK	-	VOR	-
AKT	701	NACH	-

Bild 3.24 Kettentabelle nach Zugriff auf GLIED 3. Kette nur nachfolgerverarbeitbar.

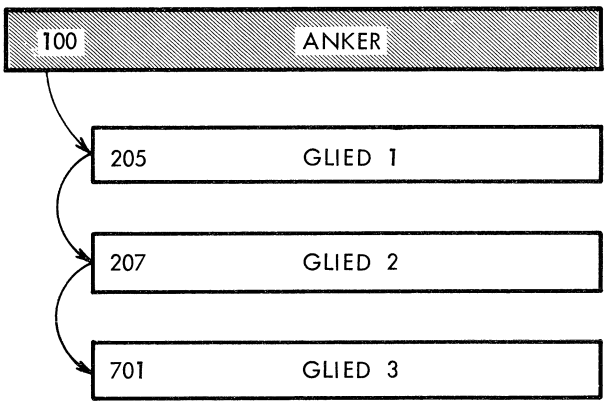


Bild 3.22 Kette mit logischen Satzadressen

Der Einsprungspunkt in die Kette sei Glied 2 und die Kette vorgängerverarbeitbar. Jetzt ist das System in der Lage, Glied 1 und von Glied 1 aus den Anker zu finden. Wie es in Bild 3.25 gezeigt wird, sieht die Kettentabelle nach dem Zugriff auf Glied 2 aus.

ANK	-	VOR	205
AKT	207	NACH	701

Bild 3.25 Kettentabelle nach Zugriff auf GLIED 2, Kette ist vorgängerverarbeitbar.

ANK	100	VOR	-
AKT	100	NACH	205

1

ANK	100	VOR	100
AKT	205	NACH	207

2

ANK	100	VOR	205
AKT	207	NACH	701

3

ANK	100	VOR	207
AKT	701	NACH	-

4

Bild 3.23 Kettentabelle, wenn ANKER (1), GLIED 1 (2), GLIED 2 (3) und GLIED 3 (4) bearbeitet wird.

Der Einsprungspunkt in die Kette sei Glied 3 und die Kette ankerverarbeitbar. Auf den Ankersatz kann somit direkt zugegriffen werden. Aber auch der Vorgänger zu Glied 3, nämlich Glied 2, ist über den Anker und Glied 1 zu finden. Nachdem auf Glied 3 zugegriffen worden ist, erscheint die Kettentabelle, wie es Bild 3.26 zeigt.

ANK	100	VOR	-
AKT	701	NACH	-

Bild 3.26 Kettentabelle nach dem Zugriff auf GLIED 3. Kette ist ankerverarbeitbar.

Der Einsprungspunkt sei Glied 2 und die Kette vorgänger- und ankerverarbeitbar. Die Kettentabelle ist nach dem Zugriff auf Glied 2 wie in Bild 3.27 besetzt.

ANK	100	VOR	205
AKT	207	NACH	701

Bild 3.27 Kettentabelle nach Zugriff auf GLIED 2. Kette ist vorgänger- und ankerverarbeitbar.

Zusammenfassend läßt sich sagen: Der Nachfolger des zuletzt bearbeiteten Satzes einer Kette ist stets zu finden, es sei denn, das Kettenende ist erreicht (DBS-Befehl: HOLEN NACHFOLGER IN Kettenname KETTE).

Der Vorgänger des zuletzt bearbeiteten Satzes einer Kette kann nur dann geholt werden, wenn die Kette anker- und vorgängerverarbeitbar ist bzw. der aktuelle Satz über den Vorgänger geholt wurde (DBS-Befehl: HOLEN VORGAENGER IN Kettenname KETTE). Während bei Vorgängerverarbeitung der Kette der Vorgänger direkt ermittelt werden kann, ist bei Ankerverarbeitung der Vorgänger nur über den Anker - durch Abarbeiten der Kette vom Anker her - zu finden.

Der Ankersatz der gerade bearbeiteten Kette kann bei einem Einsprung mitten in die Kette ebenfalls nur gefunden werden, wenn die Kette anker- oder vorgängerverarbeitbar ist (DBS-Befehl: HOLEN ANKER IN Kettenname KETTE), und zwar kann jetzt unmittelbar auf den Ankersatz zugegriffen werden, wenn die Kette ankerverarbeitbar definiert ist. Ist die Kette nur vorgängerverarbeitbar, so läuft intern ein HOLEN VORGAENGER so oft ab, bis der Ankersatz gefunden wird.

4. DBS-PROGRAMMIERUNG

4.1. Sprachelemente

4.1.1. Übersicht der Systemteile von DBS

DBS besteht aus folgenden Systemteilen:

- DBS-Programmiersprache
- DBS-Übersetzer
- DBS-Formatierer.

Darüber hinaus gibt es mehrere DBS-Utilities (DBS-Dump, DBS-Kompressor usw.), die die Einsatzmöglichkeiten von DBS erweitern und den Betrieb von DBS-Installationen komfortabler machen. Die DBS-Utilities werden im Anhang beschrieben.

Basissprachen

Unter der Basissprache wird die Höhere Programmiersprache verstanden, die in einem Datenverarbeitungssystem den verfahrensorientierten Rahmen abgibt, um eine Verarbeitung der Daten im Arbeitsspeicher zu ermöglichen. Als Basissprachen können neben COBOL auch FORTRAN, ALGOL eingesetzt werden.

Datenmanipulationssprache (DMS)

DMS ist eine Zusammenfassung aller DBS-Befehle für:

- die Systeminitialisierung
- den Datenverkehr
- den Änderungsdienst
- die Kettenverarbeitung.

4

4.1.2. DBS-Programmiersprache

DBS ist ein allgemein anwendbares System für die Organisation und Programmierung von DBS-Installationen.

DBS ist derart konzipiert, daß jeder Anwender dieses Systems mit Parametern genau seinen speziellen Anforderungen an eine Datenbank anpassen kann.

Die DBS-Programmiersprache ist eine Zusammenfassung folgender Sprachen:

Datenbankbeschreibungssprache (DBB)
Basissprache (z.B. COBOL)
Datenmanipulationssprache (DMS)

Datenbankbeschreibungssprache (DBB)

Die DBB - angewandt bei COBOL - ist eine Erweiterung des Leistungsumfangs der Statements für die Datensatz- und Felddefinition der COBOL WORKING-STORAGE SECTION durch DBS-Parameter für:

- die Gebietsbeschreibung
- die Strukturdefinition
- die Festlegung der Speicherungs- und Verarbeitungsformen.

DBB ist dabei keine eigenständige Sprache, sondern eingebettet in die COBOL DATA DIVISION und stellt eine sprachliche Erweiterung der standardisierten COBOL-Datenbeschreibung dar.

4.1.3. DBS-Übersetzer

Aus den DBS-Parametern werden vom DBS-Übersetzer Steuertabellen generiert, die den Ablauf der DBS-Befehle steuern.

Eine ausführliche Beschreibung der Steuerkarten für den Start des DBS-Übersetzers, die Ablaufphasen und die Ausgaben enthält der Teil 5 dieses Handbuches.

4.1.4. DBS-Formatierer

Der DBS-Formatierer beschreibt das Datenbankgebiet mit "leeren" Seiten, er richtet das Datenbankgebiet für eine DBS-Neuinstallation ein. Seitenlänge und Seitenzahl werden durch DBS-Parameter in der Gebietsbeschreibung festgelegt.

Der DBS-Formatierer ist im Teil 6 dieses Handbuches ausführlich beschrieben.

4.2. Datenbankbeschreibungssprache (DBB)

Im folgenden werden die DBS-Parameter vorgestellt. In den Abschnitten 4.2.1 bis einschließlich 4.2.6 wird im wesentlichen der funktionale Zusammenhang der DBB-Elemente in dem Umfang beschrieben, wie er für Organisatoren und Systemberatung von Interesse ist. Anschließend folgt eine Anleitung für die DBS-Programmierung.

4.2.1. Formaler Aufbau der DBS-Parameter

Die DBS-Parameter haben folgenden Aufbau:

* PARAMETERNAME [=] PARAMETERWERT [WAHLWORT] [{ . , ; }]

* PARAMETERNAME und PARAMETERWERT sind obligatorische Angaben.

Ein Wahlwort kann verwendet werden, um die Lesbarkeit der Datenbankbeschreibung zu erhöhen. Wahlworte sind: AM, BIS, FELD, IN, MIT, NACH, SATZ, SAETZE, SEITEN, SORTIERT, VON, ZEICHEN. Damit die Datenbankbeschreibung bereits als Programmdokumentation dienen kann, wird empfohlen, von Wahlworten möglichst weitgehend Gebrauch zu machen.

Gleichheitszeichen sind optional. Abschlußzeichen (" . ") ist obligatorisch.

Syntaktisch gelten folgende Regeln:

1. Jeder DBS-Parameter wird eingeleitet durch einen Stern (*)in Spalte 7 des Ablochformulars.Es darf nicht über mehrere Karten und nur bis Spalte 72 gelocht werden. Die DBS-Parameter: DATENBANKBESCHREIBUNG, GEBIET, DATEN, STRUKTUREN, DBS-ENDE müssen in der Area A, alle übrigen in der Area B beginnen.

2. Es muß mindestens 1 Leerzeichen zwischen den jeweiligen Angaben eines vollständig beschriebenen DBS-Parameters stehen, lediglich das Abschlußzeichen kann ohne Leerstelle folgen.

   Beispiel:
   * ANKER = TEILESTAMM SATZ.
   * GLIED = ERZEUGNISSTRUKTUR
   * EINORDNUNG SORTIERT AUFSTEIGEND NACH TEILENUMMER.

4.2.2. Gliederung der DBS-Parameter

Die Datenbankbeschreibung wird eingeleitet durch den Parameter

* DATENBANKBESCHREIBUNG.

und sie wird abgeschlossen durch

* DBS-ENDE.

Zwischen diesen beiden Parametern stehen sämtliche DBS-Parameter.

Die Datenbank wird gebietsweise beschrieben.

Die Datenbankbeschreibung wird in zwei Abschnitten durchgeführt:

1. Abschnitt: *DATENBANKBESCHREIBUNG.
   * DATENBANKNAME = Datenbankname.
   [* PASSWORT = Datenname.]
   Das Paßwort kann in der Datenbeschreibung fehlen, der Datenbankname dagegen ist unbedingt erforderlich.

2. Abschnitt: * GEBIET = Gebietsname-1.
   usw.
   * DBS-ENDE.

Jede Gebietsbeschreibung besteht aus vier Teilen:

1. Allgemeine Speichergebietsbeschreibung
2. Bereichsorganisation
3. Datenbeschreibung
4. Strukturbeschreibung

Die DBS-Parameter werden entsprechend ihrer Funktion eingeteilt in:

1. Gebietsparameter
   für die allgemeine Speichergebietsbeschreibung und die Einteilung der Gebiete in Bereiche,
2. Datenparameter
   für die Beschreibung der Datensätze und der Speicherungs- und Verarbeitungsformen,
3. Strukturparameter
   für die Beschreibung der Datenbankstruktur durch Datenketten.

Im folgenden werden die drei DBS-Parametergruppen global vorgestellt.

4.2.3. Gebietsparameter

Die Gebietsparameter werden unterschieden in globale Gebietsparameter für die allgemeine Speichergebietsbeschreibung und lokale Gebietsparameter für die Bereichsorganisation.

Mit globalen Gebietsparametern werden Angaben für das Gesamtgebiet gemacht, mit lokalen Gebietsparametern (sog. Bereichsparameter) dagegen werden nur Angaben gemacht, die sich auf einen Bereich beziehen.

Die Gebietsparameter werden eingeleitet durch den Parameter GEBIET und abgeschlossen durch den Parameter DATEN.

Globale Gebietsparamter

Die globalen Gebietsparameter beginnen mit dem Parameter GEBIET = ...

Globale Gebietsparameter sind:

GEBIET = Gebietsname.

SEITENLAENGE = Zahl-1 ZEICHEN.

Wird die Phase 1 des DBS-Übersetzers nicht gestartet, so müssen noch die Kommunikationszonen NVB und STELLVERTRETER zu den globalen Gebietsparametern eingefügt werden.
Die Satzzonen haben folgenden Aufabau:

```
01  NVB.
02  GEBIETSNAME PIC X(12) VALUE  'Gebietsname'.
02 {FEHLERCODE} {FC       } PIC 9(6) USAGE IS COMP [VALUE Zahl-1].
   {ZUGRIFF  }
02 {PUFFERZAHL} {SATZTYP  } PIC 9(6) USAGE IS COMP  [VALUE Zahl-2].
02  ANKER  PIC 9(6) USAGE IS COMP.
02 {VORGAENGER} {VOR      } PIC 9(6) USAGE IS COMP.
02 {DIREKTADRESSE} {DADER } PIC 9(6) USAGE IS COMP.
02 {NACHFOLGER} {NACH     } PIC 9 (6) USAGE IS COMP.
02  SN-ANFANG    PIC 9 (6) USAGE IS COMP.
02  SN-ENDE      PIC 9 (6) USAGE IS COMP.

01  STELLVERTRETER.
02  TYPSCHLUESSEL.
03  TYP PIC X.
03  SCHLUESSEL PIC X(max 29).
02  FILLER PIC X(6).
```

Die Länge des Feldes TYPSCHLUESSEL sollte gleich der Länge des größten ISQ-Schlüssels auf G4-Länge aufgerundet sein.

Lokale Gebietsparameter

Die lokalen Gebietsparameter werden eingeleitet durch den Parameter BEREICH = ... und werden abgeschlossen durch den nächsten Parameter GEBIET = ... bzw. durch den Parameter DATEN.

Lokale Gebietsparameter sind:

BEREICH = Bereichsname.

PASSWORT = Datenname.

LAGE = VON Seitennummer-1 BIS Seitennummer-2.

INHALT = Zahl-6 Satzname-1 SAETZE.

INHALT = Zahl-7 Satzname-2 SAETZE.

.....

.....

INHALT = Zahl-n Satzname SAETZE.

```
*GEBIET = FERTIGUNG
*     SEITENLAENGE = 2304 ZEICHEN.
*     PUFFER = 5.
*     BEREICH = STUECKLISTEN.
*     PASSWORT = ARBEITSVORBEREITUNG.
*     LAGE = VON 1 BIS 10000.
*     INHALT = 30000 TST SAETZE.
*     INHALT = 95000 EST SAETZE.
*     BEREICH = AUFTRAEGE.
*     PASSWORT = DISPOSITION.
*     LAGE = CON 100001 BIS 25000.
*     INHALT = usw.
*DATEN.
  .
  .
  .
*STRUKTUREN.
  .
  .
  .
*GEBIET = VERTRIEB.
*     SEITENLAENGE = usw.
      .
      .
      .
*DATEN.
```

Die Datenbank eines Unternehmens besteht aus den Gebieten FERTIGUNG, VERTRIEB und VERWALTUNG. Das Gebiet FERTIGUNG wird gegliedert in die Bereiche STUECKLISTEN, AUFTRAEGE usw.

Bild 4.1 Beispiel für den Parameter GEBIET

4.2.4. Datenparameter

Der Inhalt einer Datenbank umfaßt die aus Datenlementen aufgebauten Datenbestände (Dateien). Die Datenelemente werden als Datensätze in der WORKING-STORAGE SECTION definiert.

Die Datensätze, die in der Datenbank geführt werden sollen, werden zwischen den Parametern DATEN und STRUKTUREN definiert.

Jede Satzdefinition beginnt mit einem Stufenindikator 01 und dem Satznamen. Die Datenfelder dieses Satzes werden auf den Stufen 02, 03 usw. definiert.

Unmittelbar hinter jeder COBOL-Satzdefinition stehen die zu diesem Satz gehörenden Datenparameter. Dies sind:

SATZTYP = Zahl-8.

ABLAGE = { DIREKT / SEQUENTIELL / INDEX-SEQUENTIELL / NAHE Kettenname-1 KETTE / RANDOM } [MIT VERDICHTUNG].

[SCHLUESSEL = Feldname FELD.]

[VERTEILUNG = Zahl-1[, Zahl-2] .]

Je Datenbankgebiet können bis zu 127 Satzdefinitionen gegeben werden, entsprechend den 127 möglichen Satztypen je Gebiet.

Die Reihenfolge der Satzdefinitionen ist beliebig.

```
*DATEN.
 01  TST.
     02  TEILENUMMER    PIC X(12).
     02  BENENNUNG      PIC X(18).
     02  BESTAND        PIC S9(4)V99.
*    SATZTYP     = 20.
*    ABLAGE      = RANDØM.
*    SCHLUESSEL = TEILENUMMER.
 01  EST.
     02  PØS            PIC 9(3).
     02  STUECK         PIC 9(5).
*    SATZTYP     = 21.
*    ABLAGE      = NAHE STL.
 01  AUFTRAG.
     02 usw.
      :
      :
*STRUKTUREN.
```

Unter "Teil" wird allgemein ein Erzeugnis, Baugruppe, Einzelteil usw. verstanden. Für jedes Teil wird nur ein einziger Teilestammsatz (TST) und für jede Stücklistenposition ein Erzeugnisstruktursatz (EST) angelegt.

Bild 4.2 Beispiel für den Parameter DATEN.

4.2.5. Strukturparameter

Die Strukturparameter dienen der programmiertechnischen Formulierung der im Datenbank-Organigramm definierten Struktur. Die Folge der Strukturparameter wird eingeleitet durch * STRUKTUREN und beendet mit * DBS-ENDE bzw. einem Parameter GEBIET.

Kettenbeschreibungen

Strukturen werden speicherungstechnisch in Form von Adreßketten realisiert.

Jede Kettenbeschreibung wird eingeleitet durch den Parameter

* KETTE = Kettenname.

Hinter diesem Parameter stehen sämtliche Angaben für diese Kette.

Die Kettenbeschreibung wird abgeschlossen durch einen Parameter KETTE = ... GEBIET = ... bzw. durch den Parameter DBS-ENDE.

Zu jeder Kette ist nur eine Kettenbeschreibung anzugeben. Sämtliche Kettenbeschreibungen stehen unmittelbar hintereinander; die Reihenfolge ist beliebig (s. Bild 4.3).

```
*STRUKTUREN.
*    KETTE = STL.
*    ANKER = TST.
*    GLIED = EST.
*    EINØRDNUNG = SØRTIERT AUFSTEIGEND NACH PØS.
*    DUPLIKATE  = VERBØTEN.
*    ANKERWAHL  = MIT SCHLUESSEL.
*    VERKETTUNG = MIT VORGAENGER.
*    VERKETTUNG = MIT ANKER.
*    KETTE = TVN.
*    ANKER = TST.
*    GLIED = EST.
*    EINØRDNUNG = AM KETTENANFANG.
*    ANKERWAHL  = MIT TN.
*    VERKETTUNG = MIT VØRGAENGER.
*    VERKETTUNG = MIT ANKER.
 01  TN                 PIC X(12).
      :
      :
*DBS-ENDE.
```

Die Erzeugnisstrukturdaten einer Baukastenstückliste werden in der Stücklistenkette (STL), aufsteigend sortiert nach Positionsnummer (POS), NAHE dem Kettenanker gespeichert.

Der Teileverwendungsnachweis wird über die TVN-Kette geführt.

Bild 4.3 Beispiel für den Parameter STRUKTUREN.

Wird die Phase des DBS-Übersetzers nicht gestartet, so muß statt

* KETTE = ...

die COBOL-Definition:
01 Kettenname PIC X (12) VALUE "Kettenname" stehen.

4.2.6. DBS-Parameter (Übersicht)

Vollständige Übersicht und Reihenfolge der DBS-Parameter

* DATENBANKBESCHREIBUNG.

* DATENBANKNAME = Datenbankname.

[* PASSWORT = Datenname-1.]

```
* GEBIET = Gebietsname-1.
[* SEITENLAENGE = Zahl-1 ZEICHEN.]

[ 01   NVB.                                                    ]
[ 02   GEBIETSNAME        PIC X(12) VALUE 'Gebietsname',       ]
[ 02   {FEHLERCODE}       PIC 9(6) USAGE IS COMP               ]
[      {FC        }       [VALUE Zahl-1].                      ]
[      {ZUGRIFF   }                                            ]
[ 02   {PUFFERZAHL}       PIC 9(6) USAGE IS COMP               ]
[      {SATZTYP   }       [VALUE Zahl-2].                      ]
[ 02   ANKER              PIC 9(6) USAGE IS COMP.              ]
[ 02   {VORGAENGER}       PIC 9(6) USAGE IS COMP.              ]
[      {VOR       }                                            ]
[ 02   {DIREKTADRESSE}    PIC 9(6) USAGE IS COMP.              ]
[      {DADR        }                                          ]
[ 02   {NACHFOLGER}       PIC 9(6) USAGE IS COMP.              ]
[      {NACH      }                                            ]
[ 02   SN-ANFANG          PIC 9(6) USAGE IS COMP.              ]
[ 02   SN-ENDE            PIC 9(6) USAGE IS COMP.              ]

[ 01   STELLVERTRETER.                                         ]
[                                                              ]
[ 02   TYPSCHLUESSEL.                                          ]
[ 03   TYP PIC X.                                              ]
[ 03   SCHLUESSEL PIC X(max. 29).                              ]
[ 02   FILLER PIC X(6).                                        ]

* BEREICH = Bereichsname-1.
[* PASSWORT = Datenname-3.]
* LAGE = VON Seitennummer-1 BIS Seitennummer-2.
* INHALT = Zahl-2 Satzname-1 SAETZE.
[* INHALT = Zahl-3 Satzname-2 SAETZE.]
[                 :                  ]
[* INHALT = Zahl-n Satzname-n SAETZE.]
[* BEREICH = Bereichsname-2.]
[                 :                  ]
[                 :                  ]
[                 :                  ]

* DATEN.

01 Satzname-1.
   02 Feldname-11 PIC ....
   02 Feldname-12 PIC ....
   ......................
   ......................
   02 Feldname-1n PIC ....

* SATZTYP = Zahl-4.

                 {DIREKT                  }
                 {SEQUENTIELL             }
* ABLAGE = {INDEX-SEQUENTIELL             } [MIT VER-  ].
                 {NAHE Kettenname-1 ANKER }  [DICHTUNG].
                 {RANDOM                  }

[* SCHLUESSEL = Feldname-1 FELD.]
[* VERTEILUNG = Zahl-5[ , Zahl-6] .]

   01 Satzname-2.
      02 Feldname-21 PIC ....
      02 Feldname-22 PIC ....
      ......................
      ......................
      02 Feldname-2m PIC ....

* SATZTYP = Zahl-7.
   :

* STRUKTUREN.

* KETTE = Kettenname-1 SATZ.
* ANKER = Satzname-1 SATZ.
* GLIED = Satzname-2 SATZ.
* GLIED = Satzname-3 SATZ.
   ......................
   ......................
* GLIED = Satzname-n SATZ.

* EINORDNUNG =

                 {{AUFSTEIGEND}      }
                 {{ABSTEIGEND }  .... }
      {SORTIERT  {... NACH Feldname-2 FELD}}
                 {NACH TYP             }
      {AM KETTENANFANG                    }
      {AM KETTENENDE                      }.
      {DAVOR                              }
      {DANACH                             }

* ANKERWAHL = {AKTUELL                 }.
             {MIT SCHLUESSEL         }
             {MIT Feldname-3 FELD    }

[* DUPLIKATE = {VERBOTEN}.]
              {ERLAUBT }
```

```
[*  VERKETTUNG = [MIT]ANKER.          ]
[*  VERKETTUNG = [MIT]VORGAENGER.]
 [01 Feldname-3 PIC X ( ... ).]
[*  KETTE = Kettenname-2.]
          :  :
 *  GEBIET = Gebietsname-2
          :
 ......................
 ......................
 * DBS-ENDE.
```

4.2.7. DBS-Parameter

Im folgenden wird eine ausführliche Anleitung für den Gebrauch der DBS-Parameter gegeben.

Die Reihenfolge der Beschreibung der einzelnen Parameter entspricht der Gliederung der Datenbankbeschreibung.

4

INSTALLATIONSTEIL
Übersicht

* DATENBANKNAME = Datenbankname.
[* PASSWORT = Datenname-1.]
[* SYSTEM = { NEU [RESERVIERE [QUELLE]] | TEST }]

Erklärungen:

1. Zu jeder Datenbankbeschreibung gehört stets genau nur ein Installationsteil.

2. Der Installationsteil wird eingeleitet durch den Systemparameter DATENBANKNAME und abgeschlossen durch den Parameter * GEBIET.

DATENBANKNAME

Funktion

Benennung der Datenbank.

Format

* DATENBANKNAME = Datenbankname

Erklärungen:

Der Datenbankname muß eindeutig sein. Es werden nur zwölf Zeichen interpretiert. Falls gewünscht, wird unter diesem Namen in der LFD eine Datei für Versorgungsblöcke und quelle angelegt. Der Name darf keinen Bindestrich enthalten.

PASSWORT

Funktion

Sicherung der Datenbank vor unberechtigtem Zugriff.

Format

[* PASSWORT = Datenname-1] .

Erklärungen:

1. Die Datenbank kann gesichert werden durch:

   a) Paßwort für die gesamte Datenbank
   b) Paßwort für den Bereich einer Datenbank
   c) Paßwort auf Satzebene

2. Datenname-1 ist ein Feld im DBS-Arbeitsspeicher und nimmt das Kennwort auf.

Anmerkung: Die Implementierung der Paßwortbehandlung steht noch aus.

SYSTEM

Funktion

Angaben zur Erstellung der Datenbank

Format

[* System = { NEU [RESERVIERE [QUELLE]] / TEST } ,]

Erläuterungen:

Es gilt bei SYSTEM NEU:
Bei fehlerfreiem Ablauf des Übersetzers wird die Datenbank erstellt und formatiert; es werden keine Reservierungen vorgenommen.
Bei SYSTEM = NEU RESERVIERE werden lediglich die Versorgungsblöcke der GEBIETE in der LFD abgelegt, um ein späteres Einschleusen zu vereinfachen.
Bei SYSTEM = NEU RESERVIERE QUELLE wird neben den Versorgungsblöcken auch die Datenbankbeschreibung (DBS-QUELLE) abgelegt.
Bei SYSTEM = TEST wird nach fehlerfreiem Ablauf des Übersetzers und Interpretierers der Systembereich binär auf Karten ausgestanzt.
Der Parameter ist mit SYSTEM = NEU vorbesetzt.

GEBIETSTEIL
Übersicht

```
* GEBIET = Gebietsname-1.

[*      SEITENLAENGE = Zahl-1 ZEICHEN.]

[ 01    NVB.                                                    ]
[                                                               ]
[ 02    GEBIETSNAME       PIC X(12) VALUE 'Gebietsname',        ]
[ 02  { FEHLERCODE }      PIC 9(6) USAGE IS COMP                ]
[     { FC          }     [VALUE Zahl-1].                       ]
[     { ZUGRIFF     }                                           ]
[ 02  { PUFFERZAHL  }     PIC 9(6) USAGE IS COMP                ]
[     { SATZTYP     }     [VALUE Zahl-2].                       ]
[ 02    ANKER             PIC 9(6) USAGE IS COMP.               ]
[ 02  { VORGAENGER  }     PIC 9(6) USAGE IS COMP.               ]
[     { VOR         }                                           ]
[ 02  { DIREKTADRESSE }   PIC 9(6) USAGE IS COMP.               ]
[     { DADR        }                                           ]
[ 02  { NACHFOLGER  }     PIC 9(6) USAGE IS COMP.               ]
[     { NACH        }                                           ]
[ 02    SN-ANFANG         PIC 9(6) USAGE IS COMP.               ]
[ 02    SN-ENDE           PIC 9(6) USAGE IS COMP.               ]

[ 01    STELLVERTRETER.                                         ]
[                                                               ]
[ 02    TYPSCHLUESSEL.                                          ]
[ 03    TYP PIC X.                                              ]
[ 03    SCHLUESSEL PIC X(max. 29).                              ]
[ 02    FILLER PIC X(6).                                        ]

*  BEREICH = Bereichsname-1.
[* PASSWORT = Datenname-2.]
*  LAGE = VON Seitennummer-1 BIS Seitennummer-2.
*  INHALT = Zahl-2 Satzname-1 SAETZE.
[* INHALT = Zahl-3 Satzname-2 SAETZE.
   .....
   .....
*  INHALT = Zahl-n Satzname-n SAETZE.]
[* BEREICH = Bereichsname-2.
   .....
   .....                                                        ]
* DATEN.
    :
* GEBIET = ...
```

GEBIET

Funktion

Gliederung der Datenbank in Sachgebiete.

Format

* GEBIET = Gebietsname

Erklärungen:

1. Der Gebietsname muß innerhalb einer Datenbank eindeutig sein.

2. Es werden nur 12 Zeichen des Gebietsnamen interpretiert. Der Gebietsnamen selbst darf keinen Bindestrich enthalten.

3. Unter dem Gebietsnamen wird die Datenbank für das entsprechende Sachgebiet erstellt.

SEITENLAENGE

Funtion

Angabe der Seitenlänge innerhalb eines Gebietes.

Format

[* SEITENLAENGE = Zahl-1 ZEICHEN.]

Erklärungen:

1. Die Seitenlänge ist ein ganzzahliges Vielfaches der Sektorlänge des zugrundegelegten Direktzugriffsspeichers. Die TR 440-Zugriffsspeicher WSP 414 und PSP 600 haben eine Sektorlänge von 128 TR 440-Ganzworten, also von 128 x 6 = 768 Zeichen.

2. Als Seitenlänge sind z.Z. folgende Werte von Zahl-2 zugelassen:

   Zahl-1 = 128 x 6 x 1 = 768
   = 128 x 6 x 2 = 1536, (Standard)
   = 128 x 6 x 3 = 2304,
   = 128 x 6 x 4 = 3072,
   = 128 x 6 x 5 = 3840,
   = 128 x 6 x 6 = 4608,
   = 128 x 6 x 7 = 5376,
   = 128 x 6 x 8 = 6144.

3. Daten- und Indexseiten haben dieselbe Seitenlänge.

4. Fehlt der Parameter SEITENLAENGE ..., dann wird automatisch für die Seiten die Standardlänge 1536 Zeichen abgenommen.

BEREICH

Funktion

Unterteilung eines Sachgebietes in Teilgebiete

Format

[* BEREICH = Bereichsname-1.]

Erklärungen:

1. Mit dem Parameter BEREICH = ... kann der Datenteil des Datenbankgebietes unterteilt werden.

2. Der Bereichsname muß innerhalb eines Gebietes eindeutig sein; es werden nur 12 Zeichen interpretiert.

3. Der Parameter BEREICH = ... muß für jeden Satznamen gegeben werden, für den ABLAGE = SEQUENTIELL definiert wird.

PASSWORT

Funktion

Sicherung der Datenbank vor unberechtigtem Zugriff.

Format

[* PASSWORT = Datenname-1.]

Erklärungen:

1. Datenname-1 ist ein Feld im DBS-Arbeitsspeicher und nimmt das Paßwort auf.

2. Wird der Datenteil eines Datenbankgebietes nicht durch Bereiche unterteilt, dann entfällt auch das Paßwort für die Bereichssicherung.

3. Die Datenbank kann gesichert werden durch:

   a) Paßwort für die gesamte Datenbank
   b) Paßwort für den Bereich einer Datenbank
   c) Paßwort auf Satzebene.

LAGE

Funktion

Angabe der Lage eines Bereichs innerhalb eines Gebietes durch Bereichsgrenzen.

Format

[* LAGE = VON Seitennummer-1 BIS Seitennummer-2.]

Erklärungen:

1. Der Parameter LAGE = ... ist eine notwendige Ergänzung zum Parameter BEREICH = ...
2. Die Seitennummern müssen innerhalb von Teilgebiet-2 des DBS-Gebietes liegen.
3. Die Bereiche dürfen sich nicht überschneiden; der Benutzer muß selbst darauf achten, daß keine Lücken entstehen, da dies ungenützer Platz in der Datenbank ergeben würde.

INHALT

Funktion

Angabe des Namens und der Anzahl der Datensätze, die zu einem Bereich gehören.

Format

[* INHALT = Zahl-2 Satzname-1 SAETZE.]

Erklärungen:

1. Für Zahl-2 gilt:

   $0 < \text{Zahl-2} \leqq 16\,777\,215.$

2. Satzname-1 SATZ muß im Datenteil der DATENBANK-BESCHREIBUNG als Datensatz definiert sein.
3. Wird für Satzname-1 SATZ im Datenteil ABLAGE = INDEX-SEQUENTIELL mit SCHLUESSEL = ... vorgeschrieben, dann ermittelt der DBS-Übersetzer aus Zahl-6 und der Schlüssellänge automatisch die Länge des Indexbereiches und die Anzahl der Indexstufen.
4. Umgehenst muß jeder Satz vorher im Gebietsteil bei INHALT angeben worden sein.
5. Jeder Lageangabe muß mindestens eine Parameter INHALT folgen.

DATENTEIL
Übersicht

```
* DATEN.

  01  Satzname-1.
      02  Feldname-11   PIC....
      02  Feldname-12   PIC....
      ......................
      ......................
      02  Feldname-1n   PIC....

*     SATZTYP = Zahl 4.

*     ABLAGE = { DIREKT                          } [MIT          ]
               { SEQUENTIELL                    } [VERDICHTUNG  ] .
               { INDEX-SEQUENTIELL              }
               { NAHE Kettenname-1 ANKER        }
               { RANDOM                         }

[*    SCHLUESSEL = Feldname-1 FELD.]

[*    VERTEILUNG = Zahl-5 [ , Zahl-6].]

  01  Satzname-2.
      02 Feldname-21   PIC....
      02 Feldname-22   PIC....
      ......................
      ......................
      02 Feldname-2m   PIC....

*     SATZTYP = Zahl-7.
        :
```

Erklärungen

1. Zu jeder Gebietsbeschreibung gehört stets genau ein Datenteil.

2. Der Parameter DATEN leitet die Datensatzbeschreibung ein.
   Die Datensatzbeschreibung enthält mindestens eine, höchstens 127 Satzdefinitionen je Datenbankgebiet.
   Jede Satzdefinition besteht aus zwei Teilen:

   a) Definition des Satznamens auf Stufe 01,
      Definition der Datenfelder auf Stufe 02, 03 ff.

   b) Definition der Speicherungsform

3. Sämtliche Satzdefinitionen pro Gebiet folgen unmittelbar hintereinander. Die Reihenfolge der Satzdefinitionen ist beliebig.

4. Der Satzname muß eindeutig sein; es werden nur zwölf Zeichen interpretiert.

5. Die Datensatzbeschreibung wird durch den Parameter STRUKTUREN oder DBS-ENDE abgeschlossen.

SATZTYP

Funktion

Angabe einer Kennzahl, die einem bestimmten Satznamen zugeordnet wird.

Format

[* SATZTYP = Zahl-7.]

Erklärungen

1. Jedem Satznamen wird ein eindeutiger Satztyp zugeordnet.

2. Der Satztyp ist eine ganze Zahl zwischen 1 und 127 ($1 \leq$ Zahl-7 ≤ 127).

3. Der Satztyp wird vom Datenbankprozessor im Leitwort (LTW) des Datensatzes gespeichert und dient zur Klassifikation der Datensätze eines Datenbankgebietes.

ABLAGE

Funktion

Angabe der Speicherungs- und Verarbeitungsform.

Format

$$\text{ABLAGE} = \left\{ \begin{array}{l} \text{DIREKT} \\ \text{SEQUENTIELL} \\ \text{INDEX-SEQUENTIELL} \\ \text{NAHE Kettenname ANKER} \\ \text{RANDOM} \end{array} \right\} \left[\begin{array}{l} \text{MIT} \\ \text{VERDICHTUNG} \end{array} \right].$$

Erklärungen

1. ABLAGE DIREKT
   Der zu speichernde Datensatz wird in die aktuelle Seite seines Bereichs, die in der DBS-Pufferzone steht, abgelegt.

   Steht keine Seite dieses Satzbereiches in der DBS-Pufferzone, dann wird vom Datenbankprozessor die Ladeadresse von der systeminternen Speicherverwaltung ausgewählt. Vom DBS-Rahmenprogramm muß die vom Datenbankprozessor im Feld DIREKTADRESSE nach fehlerfreiem Ablauf des Befehls SPEICHERN ... hinterlegte DBS-Adresse gesichert werden, damit sie zu jedem beliebigen späteren Zeitpunkt für den Befehl HOLEN DIREKT wieder in das Feld DIREKTADRESSE eingesetzt werden kann.

2. ABLAGE SEQUENTIELL

   Für jeden starr-sequentiell zu speichernden Satztyp ist grundsätzlich ein eigener Bereich (sog. sequentieller Bereich) durch den Parameter BEREICH ... anzulegen. Der Bereich wird starr-sequentiell aufsteigend geladen; die Ladeadresse des nächsten freien Speicherplatzes wird automatisch durch die systeminterne Speicherverwaltung vorgegeben.

   Die Verarbeitungsformen für die Daten eines sequentiellen Bereichs sind sequentiell und assoziativ, falls der betreffende Satztyp Element einer Kettenstruktur ist.

3. ABLAGE INDEX-SEQUENTIELL
   Für die index-sequentiell zu speichernden Satztypen wird ein Indexbereich angelegt; der Datensatz selbst wird im Datenbereich gespeichert.

   Indexbereich
   Für die index-sequentiell zu speichernden Satztypen werden die Stellvertretersätze im Indexbereich und die dazugehörenden Datensätze im Datenbereich gespeichert.

Der Parameter ABLAGE = INDEX-SEQUENTIELL wird ergänzt durch den Parameter SCHLUESSEL ... .

4. ABLAGE NAHE
Diese assoziative Speicherungsform bedeutet, daß der betreffende Satz als Glied der durch den Kettennamen identifizierten Kette nach Möglichkeit in die physikalische Nähe des Kettenankers gespeichert werden soll.

Der Satz muß somit im gleichen Bereich liegen, wie der Ankersatz mit Kettennamen angegebenen Kette.
Sind die Bereiche verschieden, werden vom System für diesen Satz die Bereichsgrenzen des Ankers eingesetzt.

5. ABLAGE RANDOM
Die DBS-Adresse des betreffenden Satzes wir durch den DBS-Randomgenerator ermittelt.

Der Randomgenerator errechnet die Adresse aus dem Inhalt des Feldes, dessen Name der Parameter SCHLUESSEL ... angibt.

VERDICHTUNG

Funktion

Kompression eines Datensatzes.

Format

[* MIT VERDICHTUNG.]

Erklärungen

1. Der Kompressor verdichtet einen Datensatz vor dem Speichern und dehnt ihn nach dem Holen unter folgenden Bedingungen:

   a) Mindestens vier aufeinanderfolgende, gleiche Zeichen müssen entweder Leerstellen (Interncode: 'AFAFAFAF') oder Nullbytes (Interncode 'B0B0B0B0') oder positive binäre Nullen ('00000000') oder negative binäre Nullen ('FFFFFFFF') haben.

   b) Diese Gruppe gleicher Zeichen wird aus dem Satz entfernt. Der restliche Teil des Satzes wird zum Satzanfang hin verdichtet. Für jede dieser Verkürzungen wird ein Hinweis gebildet.

   c) Der durch die Verkürzung gewonnene Platz muß größer sein als der Platzbedarf für die in Punkt b erläuterten Hinweise.

2. Der Anwender muß in seinem Programm nur den Parameter VERDICHTUNG angeben. Zusätzliche Angaben oder Befehle sind nicht erforderlich.

3. Der Parameter VERDICHTUNG sollte nur da gegeben werden, wo der Gewinn an externem Speicherplatz die erhöhte Laufzeit rechtfertigt.

SCHLUESSEL

Funktion

Angabe des Datenfeldes, das den Wert für die Satzidentifizierung bei index-sequentieller bzw. bei Random-Speicherung enthält.

Format

```
[* SCHLUESSEL 3 Feldname FELD.]
```

Erklärungen

1. Der Parameter
   SCHLUESSEL = Feldname FELD.
   ist notwendig, falls für diesen Satztyp
   ABLAGE = INDEX-SEQUENTIELL.
   bzw.
   ABLAGE = RANDOM.
   vorgeschrieben ist.

2. Für sämtliche sonstigen Formen des Parameters ABLAGE ... hat der Parameter SCHLUESSEL ... keine Bedeutung.

3. Feldname FELD ist als Datenfeld des betreffenden Satzes unter Verwendung desselben Namens definiert.

VERTEILUNG

Funktion

Angabe über die Verteilung auf einzelne Seiten der index-sequentiell zu speichernden Datensätze.

Format

```
[* VERTEILUNG = Zahl-5 [ , Zahl-6].]
```

Erklärungen

1. Der Parameter VERTEILUNG ... ist eine Ergänzung zur index-sequentiellen Speicherung und hat nur Auswirkung auf einen einzigen index-sequentiell zu speichernden Satztyp innerhalb eines Bereiches.

2. Zahl-5 gibt an, in welchem Seitenabstand die Datensätze im angegebenen Bereich gespeichert werden sollen (Belegungsintervall).
   Zahl-6 gibt an, wieviele Datensätze pro Seite entsprechend dem Belegungsintervall gespeichert werden sollen. Fehlt die Zahl-6, dann wird pro n-te Seite ein Datensatz gespeichert.

3. Die Verteilung wird automatisch dann abgeschaltet, wenn durch 'SPEICHERN' das Bereichsende erreicht wurde.

4. Der Parameter VERTEILUNG ist nur dann sinnvoll in der Anwendung, wenn der Anker einer Kette index-sequentiell gespeichert wird und die Gliedsätze nahe dem Anker gespeichert werden sollen.

STRUKTURTEIL
Übersicht

```
* STRUKTUREN.
*   KETTE = Kettenname-1.
*   ANKER = Satzname-1 SATZ.
*   GLIED = Satzname-2 SATZ.
[*  GLIED = Satzname-2 SATZ.]
        :
[*  GLIED = Satzname-n SATZ ]

*   EINORDNUNG= { SORTIERT { { AUFSTEIGEND } ... } }
                { ABSTEIGEND }   ... NACH Feldname-2 FELD
                                 NACH TYP
                { AM KETTENANFANG  }
                { AM KETTENENDE    }
                { DAVOR            }
                { DANACH           }.

*   ANKERWAHL = { AKTUELL          }.
                { MIT SCHLUESSEL   }
                { MIT Feldname-3 FELD }

[*  DUPLIKATE = { VERBOTEN }.]
                { ERLAUBT  }

[*  VERKETTUNG =  MIT ANKER.]

[*  VERKETTUNG =  MIT VORGAENGER.]

 [01 Feldname-3      PIC ...]

*   KETTE = Kettenname-2
      .
      .

*   DBS-ENDE.
```

KETTE

Funktion

Kettendefinitionsindikator mit Kettenbenennung.

Format

* KETTE = Kettenname.

Erklärungen

1. Der Kettendefinitionsindikator KETTE = ... leitet die Definition einer DBS-Kette ein.

2. Der Kettenname muß eindeutig sein, er darf nicht länger als 12 Zeichen sein.

3. Der DBS-Übersetzer erstellt aus dem Parameter
   KETTE = Kettenname.
   die COBOL-Eintragung
   01 Kettenname PIC X(12) VALUE 'Kettenname'.

4. Wird die Phase 1 des DBS-Übersetzers nicht gestartet, so muß der Benutzer selbst die Kettendefinition durch die COBOL-Eintragung ersetzen.

ANKER

Funktion

Benennung des Kettenankers.

Format

* ANKER = Satzname-1 SATZ.

Erklärungen

1. Jede Kette hat stets genau einen Anker.

2. Der durch Satzname-1 SATZ benannte Satztyp muß im Datenteil der Datenbankbeschreibung unter demselben Namen auf Stufe 01 definiert sein.

GLIED

Funktion

Benennung eines Kettengliedes.

Format

```
* GLIED = Satzname-2 SATZ.
[* GLIED = Satzname-3 SATZ.
    :
*   GLIED = Satzname-n SATZ.]
```

Erklärungen

1. Zu jeder Kettendefinition gehört die Benennung von mindestens einem Satztyp, der die Glieder einer Kette stellt.

2. Die Satztypen, die durch Satzname-2, Satzname-3, ..., Satzname-n identifiziert werden, müssen untereinander und von dem durch Satzname-1 in der ANKER-Erklärung dieser Kettendefinition identifizierten Satztyp verschieden sein. Ansonsten sei auf die im Abschnitt 2.6.2 zulässigen DBS-Strukturen hingewiesen.

   Da DBS bis zu 127 Satztypen für ein Gebiet zuläßt, folgt, daß als Glieder einer Kette höchstens 126 Gliedertypen definiert werden können.

3. Durch Satzname-2, ..., Satzname-n benannten Satztypen müssen im Datenteil der Datenbankbeschreibung definiert sein.

EINORDNUNG

Funktion

Angabe, wie ein neu einzuspeichernder Datensatz in eine Kette einzugliedern ist.

Format

```
*  EINORDNUNG= { SORTIERT { { AUFSTEIGEND } ...
                          { { ABSTEIGEND  }
                            ... NACH Feldname-2 FELD
                            NACH TYP                 }
              { AM KETTENANFANG
              { AM KETTENENDE
              { DAVOR
              { DANACH                                 }
```

Erklärungen

1. EINORDNUNG = SORTIERT { AUFSTEIGEND / ABSTEIGEND } ...

   ... NACH Feldname-2 FELD

   Bedingungen

   1.1. Für die betreffende Kette ist nur ein einziger Satztyp als Gliedsatz definiert.

   1.2. Feldname ist im Arbeitsspeicher als Feld des Gliedersatzes dieser Kette definiert.

   1.3. Für diese Kette wird vom DBS-Übersetzer automatisch
   VERKETTUNG = MIT VORGAENGER
   definiert.

   Diese Form des Parameters EINORDNUNG ...
   wird ergänzt durch den Parameter DUPLIKATE ... .
   siehe Abschnitt 4.20

2. EINORDNUNG = SORTIERT NACH TYP

   Die Glieder der betreffenden Kette sind Sätze verschiedenen Typs; sie werden nach Satztyp sortiert eingekettet.

   Der Satztyp steht im Leitwort eines jeden Datensatzes. Die Sortierung nach Typ ist grundsätzlich nur aufsteigend und schließt eine weitere Sortierung nach Datenfeld (innerhalb eines Typs) aus.

3. EINORDNUNG = AM KETTENANFANG.

   Der einzuspeichernde Satz wird als erstes Kettenglied, d.h. unmittelbar hinter dem Kettenanker, eingegliedert.

4. EINORDNUNG = AM KETTENENDE.

Der einzuspeichernde Satz wird am Kettenende, d.h. als letztes Glied eingekettet.

Für diese Form der Einordnung legt der DBS-Übersetzer im Kettenanker automatisch ein Kettenfeld an, das auf das letzte Glied der Kette verweist.

5. EINORDNUNG = DAVOR

Der einzuspeichernde Satz wird unmittelbar vor dem aktuellen Satz dieser Kette als Glied eingefügt.

Ist der aktuelle Satz dieser Kette zufällig der Kettenanker, dann verfährt der Datenbankprozessor für diesen Fall so, als ob

EINORDNUNG = AM KETTENENDE

definiert worden wäre.

Für diese Form der Einordnung definiert der DBS-Übersetzer automatisch die Verkettung mit dem Vorgänger, falls für diese Verkettung nicht vom Anwender bereits vorgegeben wurde.

EINORDNUNG = DAVOR

schließt die Parameterformen

ANKERWAHL = { MIT SCHLUESSEL / MIT Feldname-3 FELD }

aus.

6. EINORDNUNG = DANACH

Der einzuspeichernde Glieder satz wird unmittelbar hinter dem aktuellen Satz dieser Kette als Glied eingefügt.

EINORDNUNG = DANACH

schließt die Parameterformen

ANKERWAHL = { MIT SCHLUESSEL / MIT Feldname-3 FELD }

aus.

ANKERWAHL

Funktion

Angabe, wie der Anker einer Kette zu finden ist, damit in diese Kette Glieder eingekettet werden können.

Format

* ANKERWAHL = { AKTUELL / MIT SCHLUESSEL / MIT Feldname-3 FELD } .

Erklärungen

1. Die Angabe

ANKERWAHL = AKTUELL

bedeutet, daß vor dem Speichern eines Kettengliedes der Anker dieser Kette als letzter Satz seines Typs vom Rahmenprogramm bearbeitet worden ist, d.h. die Ankeradresse in der Kettentabelle ist aktuell, diese Kette kann also verarbeitet werden.

Die Angabe

ANKERWAHL = AKTUELL

muß vorliegen, wenn für den Anker

ABLAGE = DIREKT, SEQUENTIELL oder NAHE Kettenname ANKER

vorgeschrieben worden ist.

2. Die Angabe

ANKERWAHL = MIT SCHLUESSEL

bedeutet, daß vom Datenbankprozessor zunächst der Kettenanker wiederaufgefunden werden muß, bevor ein Gliedsatz in die zugehörige Kette eingebaut werden kann.

Für den betreffenden Anker muß

ABLAGE = INDEX-SEQUENTIELL.
SCHLUESSEL = Feldname-1 FELD.

bzw.

ABLAGE = RANDOM.
SCHLUESSEL = Feldname-1 FELD.

vorgeschrieben sein.

Der Schlüssel für die Ankeridentifizierung muß in dem Feld stehen, das durch den Parameter

SCHLUESSEL = Feldname-1 FELD.

benannt wurde.

3. Die Angabe

ANKERWAHL = MIT Feldname-3 FELD.

bedeutet, daß unmittelbar vor dem Speichern eines Kettengliedes vom Datenbankprozessor automatisch der zugehörige Kettenanker wiederaufgefunden werden muß.

Für den betreffenden Anker muß

ABLAGE = INDEX-SEQUENTIELL.
SCHLUESSEL = Feldname-1 FELD.

bzw.

ABLAGE = RANDOM.
SCHLUESSEL = Feldname-1 FELD.

vorgeschrieben sein.

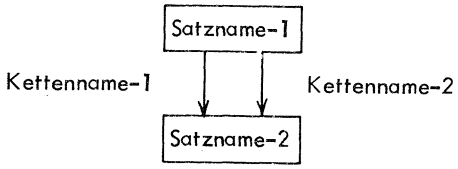
Der Schlüssel für die Ankeridentifizierung muß in das Feld mit dem Namen Feldname-3 eingesetzt werden. Dieses Feld ist ein beliebiges, unmittelbar vor der nächsten Kettendefinition im Arbeitsspeicher angegebenes Feld, das genau der Feldbeschreibung des Schlüsselfeldes entsprechen muß.

Feldname-1 und Feldname-3 müssen verschieden sein und dürfen sich nicht überlagern (z.B. durch Redefinitionen).

Die Angabe

ANKERWAHL = MIT Feldname-3 FELD

ist nur bei folgender Struktur erforderlich



Für Kettenname-1 gilt

ANKERWAHL = MIT SCHLUESSEL,

für Kettenname-2 gilt

ANKERWAHL = MIT Feldname-3 FELD.

DUPLIKATE

Funktion

Vorschrift für die Behandlung von Duplikaten in sortierten Ketten.

Format

$$\left[* \quad \underline{\text{DUPLIKATE}} = \left\{ \begin{matrix} \underline{\text{VERBOTEN}} \\ \underline{\text{ERLAUBT}} \end{matrix} \right\} . \right]$$

Erklärungen:

1. Der Parameter DUPLIKATE ... ist eine Ergänzung zum Parameter EINORDNUNG = SORTIERT NACH Feldname FELD

   Für sämtliche sonstige Formen des Parameters EINORDNUNG ... hat der Parameter DUPLIKATE ... keine Bedeutung.

2. Wenn Duplikate in einer sortierten Kette verboten sind, erfolgt eine Fehlermeldung, falls versucht wird, ein Duplikat in diese Kette einzufügen; der betreffende Satz wird nicht gespeichert. Im Feld FEHLERCODE des Nachrichtenvermittlungsblocks steht der Fehlercode als Fehlermeldung für die Auswertung durch das Rahmenprogramm, z.B. für die Erstellung eines Fehlerprotokolls (vgl. Fehlercodes im Abschnitt 7).

3. Wenn der Parameter DUPLIKATE ... fehlt, wird automatisch ein Duplikateverbot angenommen.

(VERKETTUNG)

Funktion

Angabe, ob zusätzlich zu der Verkettungsgrundform (Verkettung Nachfolger) jedes Kettenglied zusätzlich unmittelbar mit dem Kettenanker (Verkettung Anker) und/oder seinem Vorgänger (Verkettung Vorgänger) verkettet werden soll.

Format

[* VERKETTUNG = MIT ANKER.]

[* VERKETTUNG = MIT VORGAENGER.]

Erklärungen

1. Diese Zusatzverkettungen erleichtern und beschleunigen die Kettenverarbeitung.

2. Aufgrund des Parameters

   VERKETTUNG = MIT ANKER

   wird in jedem Kettenglied ein Kettenfeld angelegt, in dem die Adresse des zugehörigen Kettenankers steht. Für die Kettenverarbeitung bedeutet dies, daß von jedem Kettenglied her der Kettenanker direkt erreichbar ist.

3. Aufgrund des Parameters

   VERKETTUNG = MIT VORGAENGER

   wird in jedem Kettenglied ein Kettenfeld angelegt, in dem die Adresse seines Vorgängers steht. Für die Kettenverarbeitung bedeutet dies, daß die Kette von jedem Glied aus nicht nur "vorwärts" (im Sinne der Verkettung Nachfolger) sondern auch "rückwärts" (im Sinne der Verkettung Vorgänger) verarbeitet werden kann.

4.3. Datenmanipulationssprache (DMS)

4.3.1. Datenbankprozessor

Nachdem der Formatierer ein Datenbankgebiet eingerichtet hat, lädt der DBS-Anwender seine Datensätze in dieses Gebiet. Auf dieses Gebiet laufen die DBS-Anwendungen, die die gespeicherten Daten abfragen, verändern, löschen und neue Daten aufbereiten und in das Datenbankgebiet speichern. Dieser Auskunfts- und Änderungsdienst kann im Teilnehmer- oder im Stapelbetrieb erfolgen.

Der DBS-Anwender hat keinen unmittelbaren Zugriff auf die Datenbankgebiete. Er fordert vom Datenbankprozessor durch die Angabe eines DBS-Befehls folgende Dienstleistungen an:

1. Systeminitialisierung und -abschluß

   OEFFNEN NVB.
   ABSCHLIESSEN NVB.

2. Datentransporte

   SPEICHERN Satzname SATZ.

   HOLEN { DIREKT Satzname { SATZ / STELLVERTRETER }
   { { ANKER / VORGAENGER / NACHFOLGER } IN Kettenname KETTE
   AKTUELLER Satzname SATZ
   NACHFOLGER IN { DATENBEREICH / INDEXBEREICH }
   VORGAENGER IN INDEXBEREICH
   KETTENTABELLE AUS Kettennamen }.

3. Änderungsdienste

   AENDERN { Feldname FELD / Kettenname KETTE / Satzname SATZ }.

   LOESCHEN Satzname SATZ.

Die DBS-Befehle sind als Assembler-Unterprogramme (Makros) realisiert. Sie bilden insgesamt die Routinen im Datenbankprozessor, die die Eingabe/Ausgabe auf der logischen Stufe vorbereiten. Der DBS-Anwender arbeitet also voll rechnerunabhängig. Die Eingabe/Ausgabe auf der physikalischen Stufe führt der DBS-Input/Output-Controller (IOC) gemeinsam mit dem Betriebssystem durch.

Die Beziehungen zwischen DBS-Anwendungsprogramm (sog. Rahmenprogramm), Datenbankprozessor und Betriebssystem zeigt Bild 4.4.

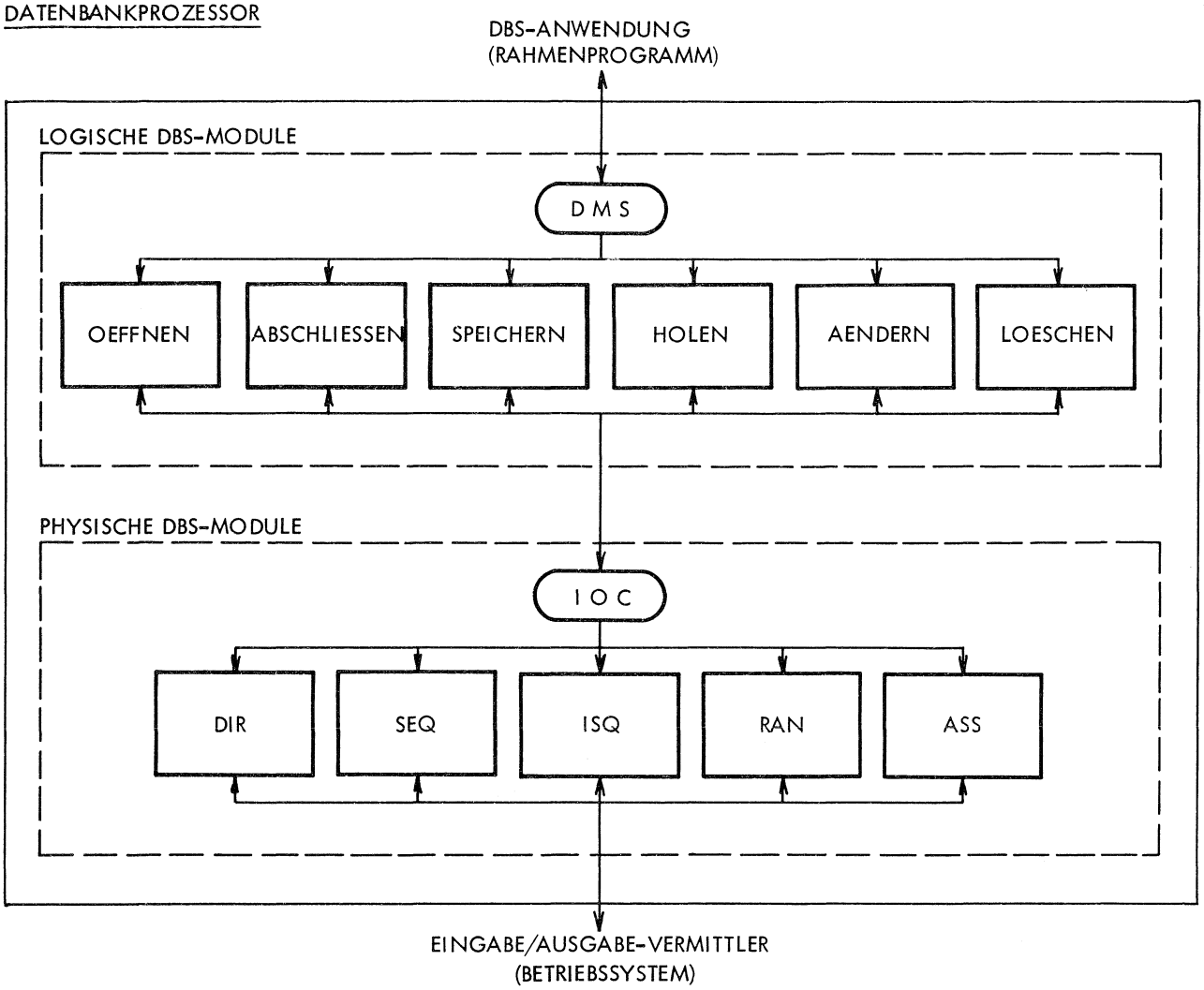


Bild 4.4 Informationsfluß während des DBS - Programmlaufs

Aus Bild 4.5 ist die Kettung der Versorgungsblöcke ersichtlich, deren Aufgabe darin besteht, eine Verbindung zu schaffen, sowohl zwischen dem Benutzerprogramm und dem Datenbankprozessor (NVB) als auch zwischen der DMS und dem IOC (SYSNVB).

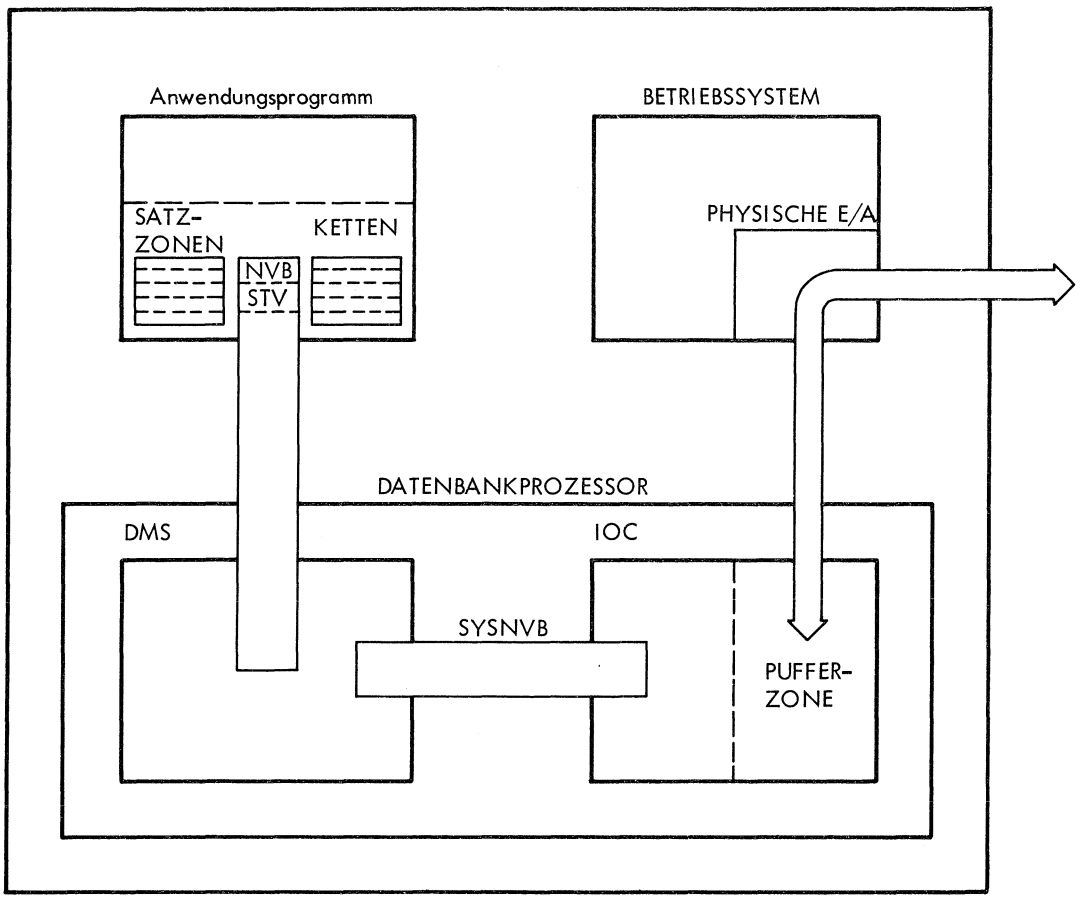


Bild 4.5 Stellung der Versorgungsblöcke

Der Datenaustausch zwischen dem Arbeitsspeicher des Rahmenprogramms und dem Datenbankspeicher verläuft nach Bild 4.6

Der Datenaustausch zwischen einem DBS-Anwenderprogramm im zentralen Arbeitsspeicher und der Datenbank erfolgt immer seitenweise. Die Seiten eines Datenbankgebietes haben stets die gleiche Länge.

Eine Seite nimmt eine Gruppe von Sätzen auf. Der erste Satz einer Seite ist stets der sog. Seitenkennsatz (SKS), er enthält neben der Seitennummer und Angaben über die Seitenbelegung Hinweise auf mögliche Überläufe usw.

Sämtliche sonstigen Sätze in einer Seite sind Datensätze. In einer Seite können bis zu 64 Datensätze gespeichert werden.

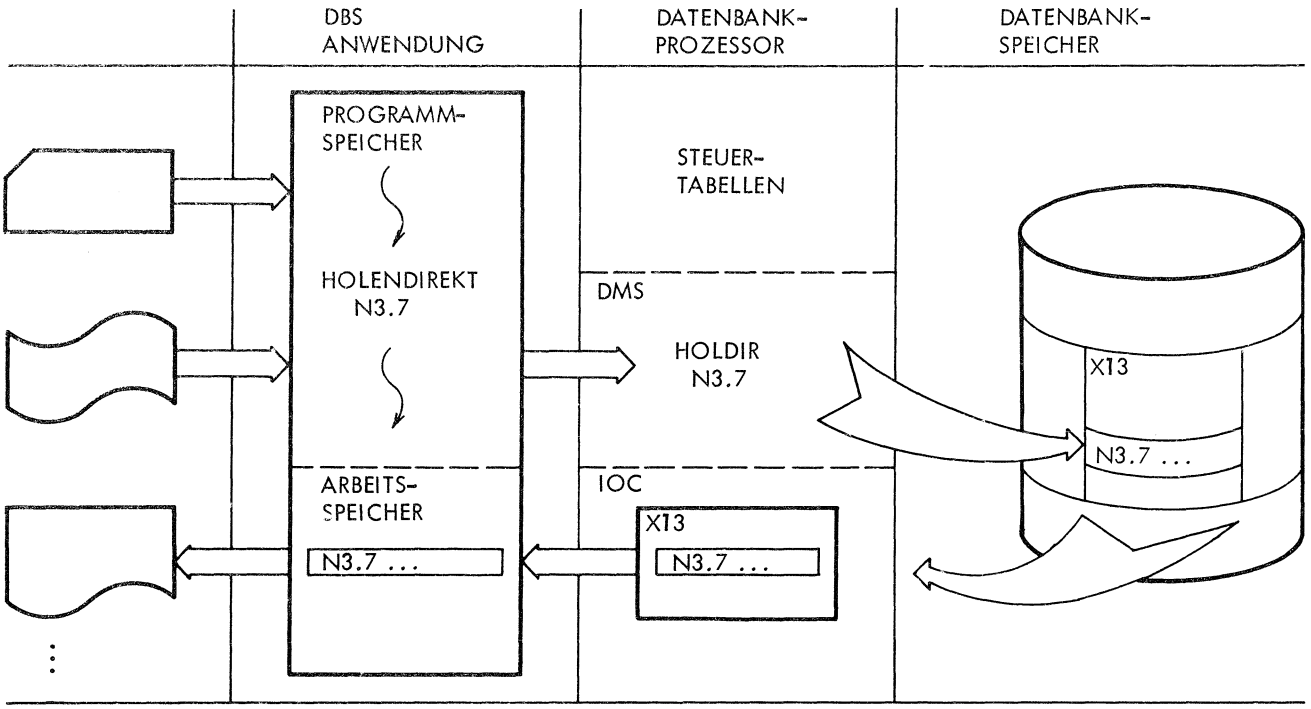


Bild 4.6 Datenaustausch zwischen Rahmenprogramm und Datenbankspeicher

4.3.2. Mnemotechnische Befehlscodes

Für die Datenbankbeschreibungssprache ist COBOL als Basissprache gewählt worden.

Für die Datenmanipulationssprache dagegen können als Basissprachen TAS, COBOL, ALGOL und FORTRAN eingesetzt werden.

Um den Befehlsaufruf und die Befehlsversorgung zu vereinheitlichen, wird die ausführliche Befehlsform ersetzt durch einen mnemotechnischen Code.

ABSCHL = ABSCHLIESSEN NVB
AENFLD = AENDERN Feldname FELD
AENKTN = AENDERN Kettenname Kette
AENSTZ = AENDERN Satzname SATZ
HOLAKT = HOLEN AKTUELLER Satzname SATZ
HOLANK = HOLEN ANKER IN Kettenname KETTE
HOLDAT = HOLEN NACHFOLGER IN DATENBEREICH
HOLDIR = HOLEN DIREKT
HOLEN = HOLEN Satzname SATZ
HOLIND = HOLEN NACHFOLGER IN INDEXBEREICH
HOLKET = HOLEN Kettentabelle AUS Kettennamen
HOLNAC = HOLEN NACHFOLGER IN Kettenname KETTE
HOLSTV = HOLEN STELLVERTRETER
HOVOIN = HOLEN VORGAENGER IN INDEX-BEREICH
HOLVOR = HOLEN VORGAENGER IN Kettenname KETTE
LOESCH = LOESCHEN Satzname SATZ
OEFFNE = OEFFNEN NVB
SPEICH = SPEICHERN Satzname SATZ

Die Befehle werden eingehend in Abschnitt 4.6 beschrieben.

4.4. DBS-Speicherplan

Der für einen DBS-Benutzer notwendige Kernspeicher gliedert sich in zwei Teile:

1. In einen für alle Anwendungsprogramme gültigen Bereich.
2. In einen vom Anwendungsprogramm abhängigen Bereich.

zu 1: In diesem Bereich werden entsprechend der in der DBS-Beschreibung festgelegten Reihenfolge die einzelnen Zonen abgelegt (siehe Bild 4.7).

- NVB (s. Abschnitt 4.4.1)
- STV (s. Abschnitt 4.4.2)
- Datensätze (s. Abschnitt 4.2.5 mit Ausnahme der mit * versehenen Eintragungen, die nur für die Erstellung der DBS-Datenbank und nicht mehr für weitere Programmläufe benötigt werden).
- Kettenname (s. Abschnitt 4.2.5, für die mit * versehenen Eintragungen gilt oben beschriebenes).

zu 2: Hier müssen die für das jeweilige Benutzerprogramm relevanten Arbeitsfelder definiert werden.

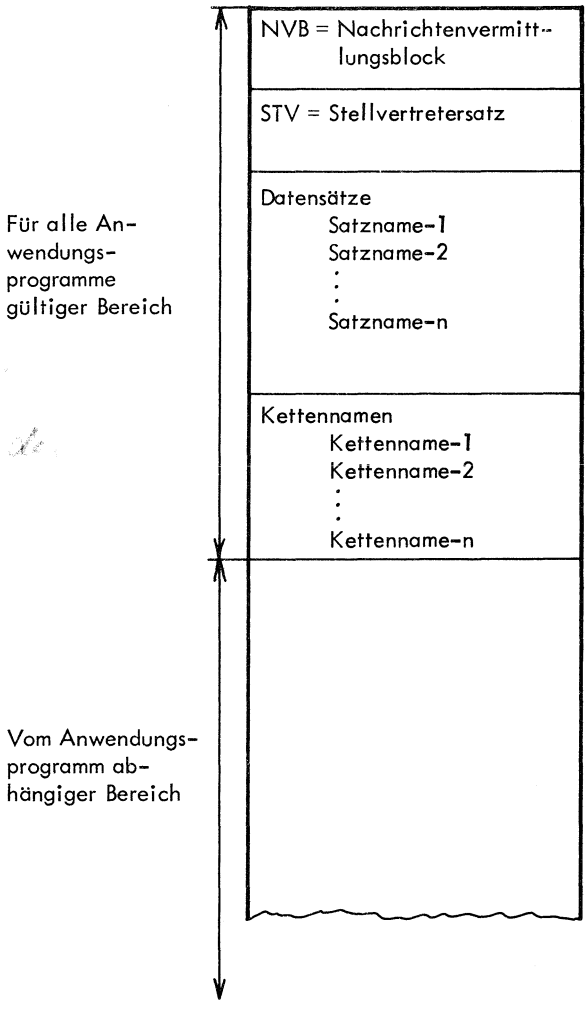


Bild 4.7 DBS - Speicherplan

4.4.1. Nachrichtenvermittlungsblock (NVB)

Der NVB dient der Kommunikation zwischen Rahmenprogramm und Datenbankprozessor. Der NVB wird im Arbeitsspeicher des Rahmenprogramms definiert. Namen, Format und Reihenfolge sind vorgeschrieben (s. Bild 4.8).

Welche Felder für die einzelnen Makros benötigt werden, ist bei der Beschreibung dieser Makros angegeben.

NVB

+0	GEBIETSNAME	
+2		
+4	ZUGRIFF bzw. FEHLERCODE	PUFFERZAHL bzw. SATZTYP
+6	ANKER	VORGAENGER
+8	DIREKTADRESSE	NACHFOLGER
+10	SN - ANFANG	SN - ENDE

Bild 4.8 Nachrichtenvermittlungsblock

Beschreibung des NVB

NVB Symbolischer Name des Nachrichtenvermittlungsblocks

NVB + 0: ..... GEBIETSNAME
Dieses Feld nimmt linksbündig den Namen desä DBS-Gebietes auf. Der Gebietsname darf nicht länger alszwölf Bytes sein und keinen Bindestrich enthalten.

NVB + 4: ..... ZUGRIFF
..... FEHLERCODE

Der Benutzer definiert im Rahmenprogramm das Feld : FEHLERCODE (bzw. FC als mnemotechnische Abkürzung). Das Feld FEHLERCODE muß beim Vielfachzugriff mit der Anzahl der gleichzeitig mit der Datenbank arbeitenden Teilnehmern für OEFFNEN vorbesetzt sein. Für diese spezielle Funktion kann er das Fled FEHLERCODE auch mit ZUGRIFF benennen.

Das Feld FEHLERCODE wird grundsätzlich vom Datenbankprozessor unmittelbar vor dem Rücksprung in das Rahmenprogramm besetzt, um dem DBS-Anwender mitzuteilen, ob der letzte DBS-Befehl normal abgelaufen ist oder ob ein Fehler entdeckt worden ist.

Normalausgang: FEHLERCODE = 0
Fehlerausgang: FEHLERCODE ≠ 0.

Die möglichen Fehlerschlüssel sind im Abschnitt 7 dieses Handbuches beschrieben.

NVB + 5: ..... PUFFERZAHL
..... SATZTYP

Wird das Feld PUFFERZAHL nicht mit einem Wert vorbesetzt, so wird vom Datenbankprozessor die standardisierte Pufferzahl 4 eingetragen. Will der Benutzer davon abweichend weniger oder mehr DBS-Puffer angeben, so muß das Feld PUFFERZAHL mit dem entsprechenden Wert vorbesetzt sein. Der DBS-Befehl OEFFNEN stellt dann die Puffer für das Objektprogramm zur Verfügung.
Nach Ablauf von OEFFNEN trägt der Datenbankprozessor in dieses Feld den SATZTYP des zuletzt bearbeitenden Satzes ein.

Nach Ablauf der Befehle

HOLEN { DIREKT / VORGAENGER / NACHFOLGER }

ist der Name des betreffenden Satzes nicht von vornherein bekannt. Das Rahmenprogramm muß daher in Abhängigkeit vom Inhalt des Feldes Satztyp verzweigen, z.B. durch

```
GO TO UP1, UP2, UP3, ...
    DEPENDING ON SATZTYP
```

falls als Satztyp die Zahlen 1, 2, 3, ... definiert sind. Ist dies nicht der Fall, dann verzweigt das Rahmenprogramm zweckmäßigerweise mit

IF ... GO TO ...

NVB + 6: ..... ANKER
In dieses Feld wird vom DBS-Prozessor die Adresse des Ankers der aktuell barbeitenden Kette eingetragen.

NVB + 7: ..... VORGAENGER
Vom DBS-Prozessor wird in dieses Feld die Vorgängerdaresse des aktuell bearbeitenden Satzes eingetragen.

NVB + 8: ..... DIREKTADRESSE
In dieses Feld trägt der Datenbankprozessor die DBS-Adresse des zuletzt geholten bzw. gespeicherten Satzes ein.

Lediglich vor dem Absetzen des Befehls
HOLEN DIREKT
setzt das Rahmenprogramm die DBS-Adresse ein.

NVB + 9: ..... NACHFOLGER
Vom DBS-Prozessor wird in dieses Feld die Nachfolgeradresse des aktuell bearbeitenden Satztes eingetragen.

NVB +10: ..... SN-ANFANG
In dieses Feld muß der Benutzer die Seitennummer der Seite eintragen, mit der das Entladen beginnen soll. Das Feld wird von HOLEN DATEN ausgewertet.

NVB + 11: ..... SN-ENDE
In dieses Feld muß vom Benutzer das Abbruchkriterium für den Befehl HOLEN DATEN eingetragen werden (natürl. Zahl von 1 ≦ DATEND).

4.4.2. Stellvertretersatz (STV)

Der Stellvertretersatz muß nur dann im Arbeitsspeicher des Rahmenprogramms angelegt werden, falls für mindestens einen Satztyp die Speicherungsform index-sequentiell definiert worden ist.

STV

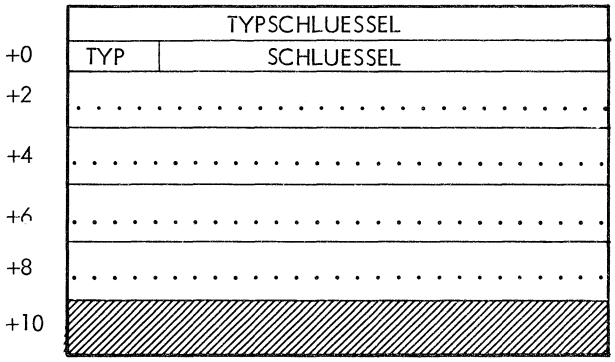


Bild 4.9 Aufbau des STV

Beschreibung des STV

STV + 0: ..... TYPSCHLUESSEL
Der Typschlüssel ist der Ordnungsbegriff, der die Sortierfolge der Stellvertretersätze in der Stellvertreterliste definiert. Der Typschlüssel besteht aus zwei Teilen:

1. TYP
Der Typ des Datensatzes, zu dem der Stellvertretersatz gehört.

Der Typ besteht im ersten Zeichen des Feldes TYPSCHLUESSEL, ist also der oberste Sortierbegriff. Daraus folgt, daß die Stellvertreter sätze sämtlicher Datensätze eines Typs logisch unmittelbar aufeinander folgen.

2. SCHLUESSEL
Schlüssel des Datensatzes, zu dem der Stellvertretersatz gehört.

Das Schlüsselfeld eines index-sequentiell zu speichernden Satztyps wird durch den Parameter SCHLUESSEL ... festgelegt.

Die maximale Länge des Feldes SCHLUESSEL ist 29 Zeichen.

4.5 Argumentlisten

Die Routinen der DMS, die die Dienstleistungen erbringen sollen, die vom Rahmenprogramm angefordert werden, sind in TAS (Telefunken-Assembler-Sprache) programmiert. Unabhängig davon, ob das Rahmenprogramm in COBOL, ALGOL, FORTRAN oder TAS geschrieben ist, sind die Befehlsroutinen einheitlich mit Argumenten zu versorgen.

In DBS/COBOL werden die DBS-Befehle über ENTER... aufgerufen und mit USING ... versorgt.

In DBS/ALGOL werden die DBS-Befehle und die Argumente als Codeprozeduren vereinbart.

In DBS/FORTRAN werden die DBS-Befehle über CALL Befehlsname (Argument) aufgerufen.

4.5.1. DBS/TAS

Der Aufruf von DBS-Befehlen in DBS/TAS erfolgt durch einen einfachen Sprungbefehl (SFB) auf die jeweils angeforderte Routine im Datenbankprozessor.

4.5.2. DBS/COBOL

Die DBS-Befehle stehen in der PROCEDURE DIVISION des DBS/COBOL-Programms mitten unter den COBOL-Befehlen, und zwar in der Befehlsfolge an genau den Stellen, an denen die zu lösende Aufgabe den Verkehr mit der Datenbank erforderlich macht.

Den Übergang von COBOL nach DBS markiert die Eintragung ENTER TAS.

Die Versorgung der DBS-Befehle erfolgt mit USING ... (vgl. TR 440-COBOL-Handbuch und Bild 4.10).

```
PROCEDURE DIVISION.
        .
        .
        .
    ENTER TAS
        OEFFNE USING NVB.
        .
        .
    IF FEHLERCODE NOT EQUAL ZERO
        GO TO ABBRUCH.
        .
        .
        .

    ENTER TAS
        HOLEN USING TST.
    IF FEHLERCODE NOT EQUAL ZERO
        GO TO FEHLERAUSGANG.
        .
        .
        .
        .
    ENTER TAS
        ABSCHL USING NVB.
    IF FEHLERCODE NOT EQUAL ZERO
        GO TO FEHLERAUSGANG.
```

Bild 4.10 Aufruf aus COBOL

Nach jedem abgesetzen DBS-Befehl muß das Feld FEHLER-CODE im NVB abgefragt werden.

4.5.3. DBS/ALGOL

Ausführliche Beschreibung folgt.

4.5.4. DBS/FORTRAN

Ausführliche Beschreibung folgt.

4.6 DBS-Befehle

Im folgenden werden in alphabetischer Reihenfolge die DBS-Befehle beschrieben.

ABSCHLIESSEN

Funktion

Alle Seiten, die nach dem Holen im Kernspeicher verändert worden sind und noch im Kernspeicher stehen, werden auf den Datenbankträger zurückgeschrieben.

Format

ABSCHL USING NVB.

Erklärungen

1. Der Befehl ABSCHL muß als letzter DBS-Befehl eines jeden DBS-Programms ausgeführt werden.

2. Der NVB muß in den ersten zwölf Zeichen den Gebietsnamen enthalten.

AENDERN
Übersicht

Funktion

Ändern von einzelnen Feldern oder eines ganzen Satzes bzw. umketten eines Satzes an einen anderen Anker.

Format

```
AEN { FLD } USING {Feldname-1 [ , Feldname-2] ....}
    { STZ }       {Satzname                      }
    { KTN }       {Kettenname                    }
```

Erklärungen

1. Ändern bezieht sich auf den unmittelbar zuletzt bearbeiteten Datensatz.

2. Tritt beim Ändern ein Fehler auf, so wird dieser Fehler dem Benutzer im Feld FEHLERCODE im NVB mitgeteilt und der alte Zustand wiederhergestellt.

AENDERN FELDNAME

Funktion

Ändern von Feldinhalten des aktuellen Datensatzes.

Format

AENFLD USING Feldname-1 [, Feldname-2] .

Erklärungen

1. Feldname-1, Feldname-2 ... müssen innerhalb des zuletzt bearbeiteten Satzes definiert sein.

2. Der Inhalt des angegebenen Feldes im Arbeitsspeicher wird in das entsprechende Feld in der DBS-Pufferzone eingesetzt.

3. Feldname-1, Feldname-2 ... können Sortierfelder, Schlüsselfelder oder beides zugleich sein.

4. Ist das zu ändernde Feld ein Sortierfeld, dann wird der Satz aus der alten Sortierfolge herausgenommen und entsprechend dem neuen Sortierbegriff in seine Kette wieder eingegliedert.

5. Ist das zu ändernde Feld ein Schlüsselfeld, dann wird in Abhängigkeit von der Speicherungsform random bzw. index-sequentiell der Satz entsprechend dem neuen Ordnungsbegriff in eine neue Randomkette eingehängt bzw. es werden im Indexbereich entsprechende Änderungen durchgeführt.

AENDERN SATZNAME

Funktion

Ändern des aktuellen Datensatzes.

Format

AENSTZ USING Satzname.

Erklärungen

1. Der Satzname muß auf der Stufe 01 im Datenteil der Datenbankbeschreibung definiert worden sein.

2. Der alte Satz wird durch den im Arbeitsspeicher bereitgestellten neuen Satz überschrieben.

3. Sind die Inhalte von Sortier- und Schlüsselfeldern in dem Satz geändert worden, dann wird der Satz aus der alten Sortierfolge ausgekettet und entsprechend dem neuen Sortierbegriff wieder eingeordnet bzw. der Satz wird in Abhängigkeit von der Speicherungsform entsprechend dem neuen Ordnungsbegriff in eine neue Randomkette eingekettet bzw. es werden im Indexbereich entsprechende Änderungen durchgeführt.

AENDERN KETTENNAME

Funktion

Umketten eines Satzes zu einem anderen Anker der Kette gleichen Namens.

Format

AENKTN USING Kettenname.

Erklärungen

1. Der Kettenname muß im Strukturteil der Datenbankbeschreibung definiert worden sein.

2. Es wird der zuletzt bearbeitete Satz aus seiner alten Kette herausgelöst und in eine andere Kette des gleichen Kettennamens wieder eingekettet.

3. Ist die Kette nur nachfolgerverarbeitbar, muß der Satz, um ihn ausketten zu können, über den logischen Vorgänger der Kette (d.h. praktisch über den Anker der Kette) wiederaufgefunden worden sein.

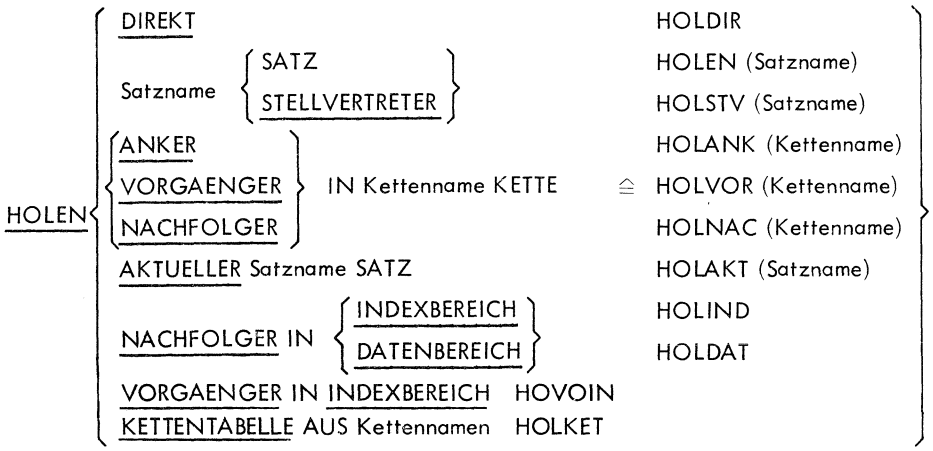
4. Hat der Anker der Kette die ABLAGE = RANDOM bzw. INDEX-SEQUENTIEL, erwartet DBS in dem entsprechenden Schlüsselfeld des Arbeitsspeichers den Ordnungsbegriff des neuen Ankers, in dessen Kette der aktuelle Satz eingekettet werden soll.

5. Hat der Anker der Kette eine andere Speicherungsform, dann muß die DBS-Adresse des neuen Ankers im Feld DIREKTADRESSE des NVB bereitgestellt werden.

HOLEN
Übersicht

Funktion

Wiederauffinden eines Satzes aus der Datenbank und Bereitstellen im Arbeitsspeicher.



Erklärungen

1. Die Satznamen müssen auf der Stufe 01 im Datenteil der Datenbankbeschreibung definiert sein.

2. Der Kettenname muß im Strukturteil der Datenbankbeschreibung definiert sein.

3. Nach erfolgreichem Suchvorgang wird
   - der Satz in den Arbeitsspeicher übertragen
   - der Typ des gefundenen Satzes im Feld SATZTYP und seine logische Adresse im Feld DIREKTADRESSE des Nachrichtenvermittlungsblocks eingetragen.
   - die Kettentabellen werden auf den neuesten Stand gebracht.

4. Der zur Verfügung gestellte Datensatz ist der aktuelle Satz seines Typs und jeder Kette, in der er Anker oder Gliedsatz ist.

5. Wenn ein Satz nicht gefunden werden kann, wird ein Fehler im Feld FEHLERCODE des Nachrichtenvermittlungsblocks gemeldet.

HOLEN AKTUELL

Funktion

Holen des zuletzt verarbeiteten Satzes vom angegebenen Typ.

Format

HOLAKT USING Satzname.

Erklärungen

1. Der zur Verfügung gestellte Satz ist der zuletzt verarbeitete Satz des durch den Satznamen bezeichneten Satztyps.

2. Wurde kein Satz dieses Satznamens verarbeitet oder wurde der zuletzt bearbeitete Satz mit diesem Satznamen gelöscht, wird ein Fehler im Feld FEHLERCODE gemeldet.

3. Nach erfolgreichem Suchvorgang wird
   - der Satz in den Arbeitsspeicher übertragen
   - der Typ des gefundenen Satzes im Feld SATZTYP und seine logische Adresse im Feld DIREKTADRESSE des Nachrichtenvermittlungsblocks eingetragen.
   - die Kettentabellen werden auf den neuesten Stand gebracht.

HOLEN ANKER

Funktion

Holen eines Ankersatzes einer Kette.

Format

HOLANK USING Kettenname.

Erklärungen

Dieser Befehl kann nur benutzt werden, wenn für die Kette VERKETTUNG MIT ANKER oder MIT VORGAENGER definiert und bereits irgendein Satz der angegebenen Kette verarbeitet worden ist oder wenn früher während der Verarbeitung dieser Kette auf den Anker zugegriffen worden ist und seither kein Ankerwechsel erfolgt ist.

HOLEN IN DATENBEREICH

Funktion

Entladen einer Folge von Datenseiten.

Format

HOLDAT.

Erklärungen

1. Die Seitennummer der Seite, mit der das Entladen beginnen soll, ist in SN-ANFANG anzugeben. Die Seitennummer der Seite, mit der das Entladen der Datensätze enden soll, ist in SN-ENDE zu übergeben.

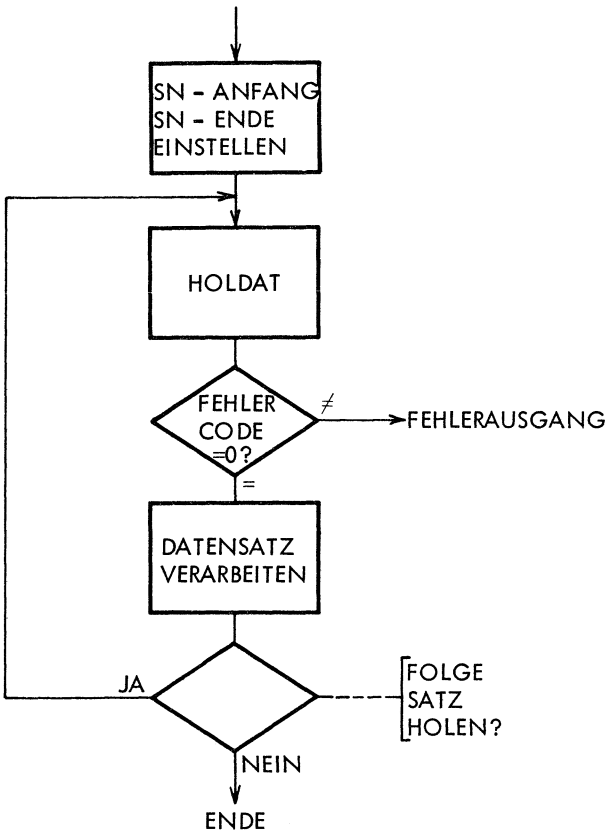
   Die Datensätze werden ab SN-ANFANG bis einschließlich SN-ENDE zur Verfügung gestellt.

   SN-ANFANG und SN-ENDE sind Felder im Nachrichtenvermittlungsblock.

2. Ist kein Datensatz mehr vorhanden, so wird dies dem Benutzer im Feld FEHLERCODE im Nachrichtenvermittlungsblock mitgeteilt.

3. Eine Verarbeitungsphase von HOLDAT, die durch einen Fehlercode ≠ 0 abgeschlossen wird, darf nicht durch einen anderen DBS-Befehl unterbrochen werden.

Anwendungsbeipiel



HOLEN DIREKT

Funktion

Holen eines Satzes, dessen logische Satzadresse bekannt ist.

Format

HOLDIR.

Erklärungen

Es wird derjenige Satz zur Verfügung gestellt, dessen logische Satzadresse der Benutzer im Feld DIREKTADRESSE des Nachrichtenvermittlungsblockes bereitgestellt hat.

HOLEN

Funktion

Holen eines index-sequentiell oder random gespeicherten Datensatzes.

Format

HOLEN USING Satzname.

Erklärungen

1. Es wird der Satz zur Verfügung gestellt, dessen Ordnungsbegriff in dem durch SCHLUESSEL = ... gekennzeichneten Feld des Arbeitsspeichers bereitgestellt wurde.
2. Nach erfolgreichem Suchvorgang wird
   - der Satz in den Arbeitsspeicher übertragen
   - der Typ des gefundenen Satzes im Feld DIREKADRESSE des Nachrichtenvermittlungsblocks eingetragen.
   - die Kettentabellen werden auf den neuesten Stand gebracht.

HOLEN KETTENTABELLE

Funktion

Holen der Kettentabelle des angegebenen Kettennamens.

Format

HOLKET USING Kettenname.

Erklärungen

1. Die Kettentabelle der mit dem Kettennamen angegebenen Kette wird in die Felder ANKER, VORGAENGER, DIREKTADRESSE und NACHFOLGER des Nachrichtenvermittlungsblocks übertragen.

HOLEN IN INDEXBEREICH

Dieser Befehl ermöglicht eine logisch-sequentielle Verarbeitung des Datenbestandes ohne vorherige Sortierung.

Funktion

Holen einer Folge von Stellvertretersätzen.

Format

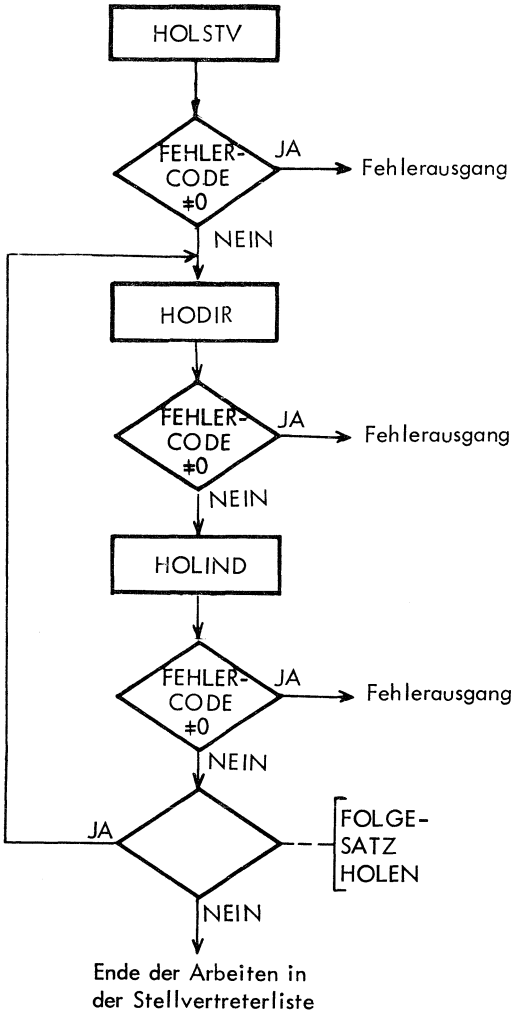
HOLIND.

Erklärungen

1. Der erste Satz einer Folge von Stellvertretersätzen ist durch HOLEN Satzname STELLVERTRETER (HOLSTV) zu holen. Die Nachfolger dieses Satzes in der Stellvertreterliste werden durch HOLEN IN INDEXBEREICH geholt.

2. Es wird immer der Stellvertretersatz mit dem nächsthöheren Ordnungsbegriff im Feld STELLVERTRETER des Arbeitsspeichers zur Verfügung gestellt. Die logische Adresse des zugehörigen Datensatzes wird in DIREKTARDESSE übergeben.

3. Ist die Stellvertreterliste zu Ende, so wird dies dem Benutzer im Feld FEHLERCODE durch Fehlerschlüssel = 006 mitgeteilt.



HOLEN NACHFOLGER

Funktion

Holen des Nachfolgersatzes in einer Kette.

Format

HOLNAC USING Kettenname.

Erklärungen

Es wird in der mit Kettenname bezeichneten Kette der Nachfolger des gerade bearbeiteten Satzes geholt. Ist kein Nachfolger vorhanden oder ist das Kettenende erreicht, wird dies dem Benutzer durch die Zahl 6 in FEHLERCODE mitgeteilt.

HOLEN STELLVERTRETER

Funktion

Holen eines Stellvertretersatzes.

Format

HOLSTV USING Satzname.

Erklärungen

1. HOLEN Satzname STELLVERTRETER kann nur gegeben werden, wenn für den zugehörigen Datensatz ABLAGE = INDEX-SEQUENTIELL definiert worden ist.

2. Der Ordnungsbegriff des gewünschten Stellvertretersatzes ist im Schlüsselfeld des unter Satzname definierten Satzes bereitzustellen. Ist für den angegebenen Ordnungsbegriff kein Stellvertreter vorhanden, wird der Stellvertreter mit dem nächsthöheren Ordnungsbegriff zur Verfügung gestellt. Diese Form ermöglicht das Wiederauffinden von index-sequentiell gespeicherten Sätzen ohne den genauen Einsprungspunkt zu kennen.

3. Die logische Adresse des zu dem geholten Stellvertretersatz gehörenden Datensatzes wird im Feld DIREKT-ADRESSE der dazugehörende Satztyp im Feld SATZTYP des Nachrichtenvermittlungsblocks übergeben. Durch HOLEN DIREKT kann der Datensatz sofort geholt werden.

4. Nach dem Befehl HOLEN Satzname STELLVERTRETER kann eine Folge von Befehlen HOLEN IN INDEXBEREICH abgegeben werden.

5. Nach erfolreichem Suchvorgang wird
   - der Satz in den Arbeitsspeicher übertragen
   - der Typ des gefundenen Satzes im Feld SATZTYP und seine logische Adresse im Feld DIREKTADRESSE des Nachrichtenvermittlungsblocks eingetragen.
   - die Kettentabellen werden auf den neuesten Stand gebracht.

HOLEN VORGAENGER

Funktion

Holen des Vorgängersatzes in einer Kette.

Format

HOLVOR USING Kettenname.

Erklärungen

1. Dieser Befehl kann nur benutzt werden, wenn
   - bereits irgendein Gliedsatz der angegebenen Kette verarbeitet worden ist und
   - für die Kette VERKETTUNG MIT ANKER oder MIT VORGAENGER definiert worden ist.

2. Ist für die Kette weder VERKETTUNG MIT VORGAENGER noch VERKETTUNG MIT ANKER definiert, dann kann höchstens nur der zuletzt bearbeitete Vorgänger geholt werden.

3. Der Vorgänger ist jener Satz, der dem gerade bearbeiteten Gliedsatz der Kette logisch vorangeht.

4. Ist der Vorgänger der Anker in einer Kette, dann wird der Ankersatz in der entsprechenden Satzzone bereitgestellt und Fehlercode 10 im Feld FEHLERCODE übergeben.

HOLEN VORGAENGER IN INDEXBEREICH

Funktion

Holen einer Folge von Stellvertretersätzen

Format

HOVOIN.

Erklärungen

1. Der erste Satz einer Folge von Stellvertretersätzen ist durch HOLEN Satzname STELLVERTRETER (HOLSTV) zu holen. Die Vorgänger dieses Satzes in der Stellvertreterliste werden durch HOLEN VORGAENGER IN INDEXBEREICH geholt.

2. Es wird immer der Stellvertretersatz mit dem nächstniedrigeren Ordnungsbegriff im Arbeitsbereich STELLVERTRETER zur Verfügung gestellt. Der Satztyp wird nach SATZTYP und die logische Adresse des zugehörigen Datensatzes wird nach DIREKTADRESSE der Nachrichtenvermittlungszone (NVB) übertragen.

3. Ist der Anfang der Stellvertreterliste erreicht, dann wird dies dem Benutzer durch den Fehlerschlüssel = 006 im Feld FEHLERCODE angezeigt.

LOESCHEN

Funktion

Löschen des zuletzt bearbeitenden Satzes und Löschen aller von ihm abhängiger Gliedersätze. Außerdem wird der Satz aus allen Ketten entfernt, in denen er Gliedsatz ist.

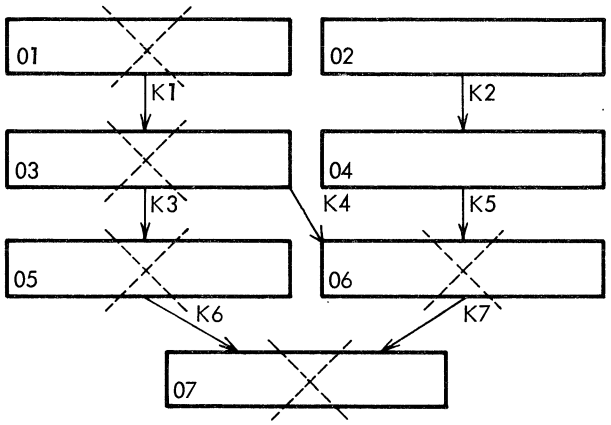
Format

LOESCH USING Satzname.

Erklärungen

1. Nach der Ausführung des Befehls LOESCHEN ist der Satz nicht mehr verfügbar. Beim Versuch, auf einen solchen Satz zuzugreifen, wird ein Fehler im Feld FEHLERCODE gemeldet.

2. Bei dem Löschvorgang werden mit dem zu löschenden Satz alle in der Kettenhierarchie unter ihm stehenden Sätze gelöscht. Angefangen wird auf der untersten Stufe der Struktur.



Beispiel: Löschen eines Satzes vom Typ 01

OEFFNEN

Funktion

Öffnen eines Datenbankgebietes für ein DBS-Programm.

Format

OEFFNE USING NVB.

Erklärungen

1. Der Befehl OEFFNEN muß als erster DBS-Befehl gegeben werden. Der Befehl OEFFNEN lädt aus dem Systembereich die DBS-Steuertabellen, initialisiert die Pufferzone usw.

SPEICHERN

Funktion

Der Datensatz wird in die Datenbank eingespeichert, alle erforderlichen Verkettungen werden durchgeführt.

Format

SPEICH USING Satzname.

Erklärungen

1. Der Satzname muß auf der Stufe 01 im Datenteil der Datenbankbeschreibung definiert sein.

2. Vor dem Absetzen des Befehls SPEICHERN ist folgendes zu beachten:
   - Der zu speichernde Satz ist in dem Arbeitsspeicher aufzubauen.
   - Bei ANKERWAHL MIT SCHLUESSEL oder ANKERWAHL MIT Feldname FELD sind alle Schlüsselfelder jener Ankersätze, die dem Satz in seiner Kettenhierarchie übergeordnet sind, zu initialisieren.
   - Bei ANKERWAHL AKTUELL muß vom Benutzer der entsprechende Ankersatz für den speichernden Gliedsatz zur Verfügung gestellt werden .

3. Der zu speichernde Satz wird entsprechend den Ablagevorschriften in der Satzbeschreibung und der entsprechenden Beschreibung im Strukturteil gespeichert.

4. Nach erfolreicher fehlerfreier Speicherung steht im Feld DIREKTADRESSE des Nachrichtenvermittlungsblockes die logische Adresse des eingespeicherten Satzes zur Verfügung.

5. Ist beim Speichern ein Fehler aufgetreten, so wird dieser im Feld FEHLERCODE angeziegt und es wird der Zustand in der Datenbank hergestellt, der vor dem Aufruf von SPEICHERN bestand.

Der DBS-Übersetzer arbeitet in zwei Phasen, um die DBS-Parameter der Datenbankbeschreibungssprache in der WORKING-STORAGE SECTION des DBS/COBOL-Rahmenprogramms auszuwerten (vergl. Bild 5.1).

Die erste Phase wird mit PROGRAMM = PHA1 im STARTE-Kommando gestartet, hinter der Spezifikation DATEN=/ folgt das COBOL-Programm.

Der DBS-Übersetzer fügt in dieser Phase im Installationsteil die Versorgungblöcke NVB (Nachrichtenvermittlungsblock) und STV (Stellvertreter) ein, die der Kommunikation zwischen dem DBS-Programm und dem Datenbankprozessor dienen. Die Datenfelder dieser Versorgungsblöcke können wie alle Felder des Arbeitsspeichers vom DBS-Programm angesprochen werden.

Weiter werden im Strukturteil die Parameter *KETTE = Kettenname KETTE ersetzt durch ein elementares Datenfeld, das folgendes Aussehen hat:

```
01 Kettenname PIC X(12) VALUE 'Kettenname'.
```

Nach Beendigung der Phase1 steht die geänderte COBOL-Quelle in der Datei DBS&QUELLE, die vor dem Start der Phase2 übersetzt werden muß. In der COBOL-UEBERSETZE-Karte muß VERSION = DS angegeben sein. Daraufhin erstellt der COBOL-Compiler für die zweite Phase des DBS-Übersetzers eine Datei mit den COBOL-Adreßbüchern.

Mit einer entsprechenden Steuerkarte (s. Bild 6.1) startet der Benutzer die zweite Phase des DBS-Übersetzers.

Dieser zweite DBS-Übersetzerlauf nimmt die Übersetzung und Interpretation der DBS-Parameter vor.

Er prüft die Parameter auf Vollständigkeit sowie syntaktische und logische Richtigkeit ab und erstellt daraus die für DBS erforderlichen Systemsteuertabellen.

Ist der Übersetzungsvorgang fehlerfrei abgelaufen, werden folgende zusätzlichen Berechnungen gemacht:

- Berechnen der Seitennummern (SN) für Anfang und Ende der Daten und Indexbereiche
- Länge der Systembereiche
- Anzahl der benötigten Indexstufen
- Länge der gesamten Datenbank in Blöcken zu 128 GW (TR 440-Achtelseiten).

UEB4 SPR.=COBOL

STARTE, DBS UEB2

Fehleranzahl nach Ablauf des Übersetzers = 0.

Ist der Syntaxlauf des DBS-Übersetzers fehlerfrei abgelaufen, wird anschließend eine Formatierung der zu erstellenden Datenbank vorgenommen, falls der DBS-Parameter SYSTEM = NEU vorliegt. Durch den Formatierer werden alle Seiten des Daten- und Indexbereiches mit entsprechenden Seitenkennsätzen versehen.

Dabei werden die Index- und Stellvertreterseiten
'FFF FFF FFF FFF' /3 und die Datenseiten
'AF AF ... AF' /3 (Leerstellen) beschrieben.

* SYSTEM = NEU [RESERVE [QUELLE]].

Eine Formatierung des Datenbankgebietes wird nur dann vorgenommen, wenn vom Benutzer SYSTEM = NEU angegeben wurde.

Sind auch die weiteren Spezifikationen RESERVIERE und QUELLE angegeben, so wird eine weitere Datei angelegt, in der für spätere Datenbankbearbeitungen wichtige Informationen und die DBS-Datenbankbeschreibung abgelegt wird.

Der Benutzer braucht somit in seinen COBOL-Programmen, die mit der bereits erstellten Datenbank arbeiten, die DBS-Datenbankbeschreibung nicht mehr explizit anzugeben, da diese dann automatisch in der ersten Phase des DBS-Übersetzers von der Datenbank eingelesen und in die COBOL-Quelle eingefügt wird, wenn dort der Parameter
* DBS-COPY = Datenbankname angegeben wurde
und im Datenbankerstellungslauf die DBS-Datenbankbeschreibung wie oben beschrieben abgelegt wurde.

Der Parameter ist mit SYSTEM = NEU
vorbesetzt.

* SYSTEM = TEST (siehe 8.2.2)

Bei der Angabe SYSTEM = TEST werden bei fehlerfreiem Syntaxlauf die DBS-Steuertabellen nicht in den Systembereich eingetragen, sondern auf Binärkarten gestanzt. Der Benutzer braucht in diesem Fall nicht nochmals einen Übersetzungslauf vorzunehmen.

Soll das DBS/COBOL-Programm gestartet werden, so sind vor die COBOL-Quelle eine Steuerkarte für den Formatierer (der in diesem Fall als eigener Operatorlauf im System vorhanden ist) und die Binärkarten zu legen.

Der Formatierer liest die auf Binärkarten vorliegenden Systemsteuertabellen ein und nimmt eine Formulierung der Datenbank vor.

Danach kann ein Start des DBS/COBOL-Programms erfolgen.

Die Reihenfolge der Steuer- und Quellkarten für das TR 440-Teilnehmer-Betriebssystem BS3 zeigt Bild 6.1.

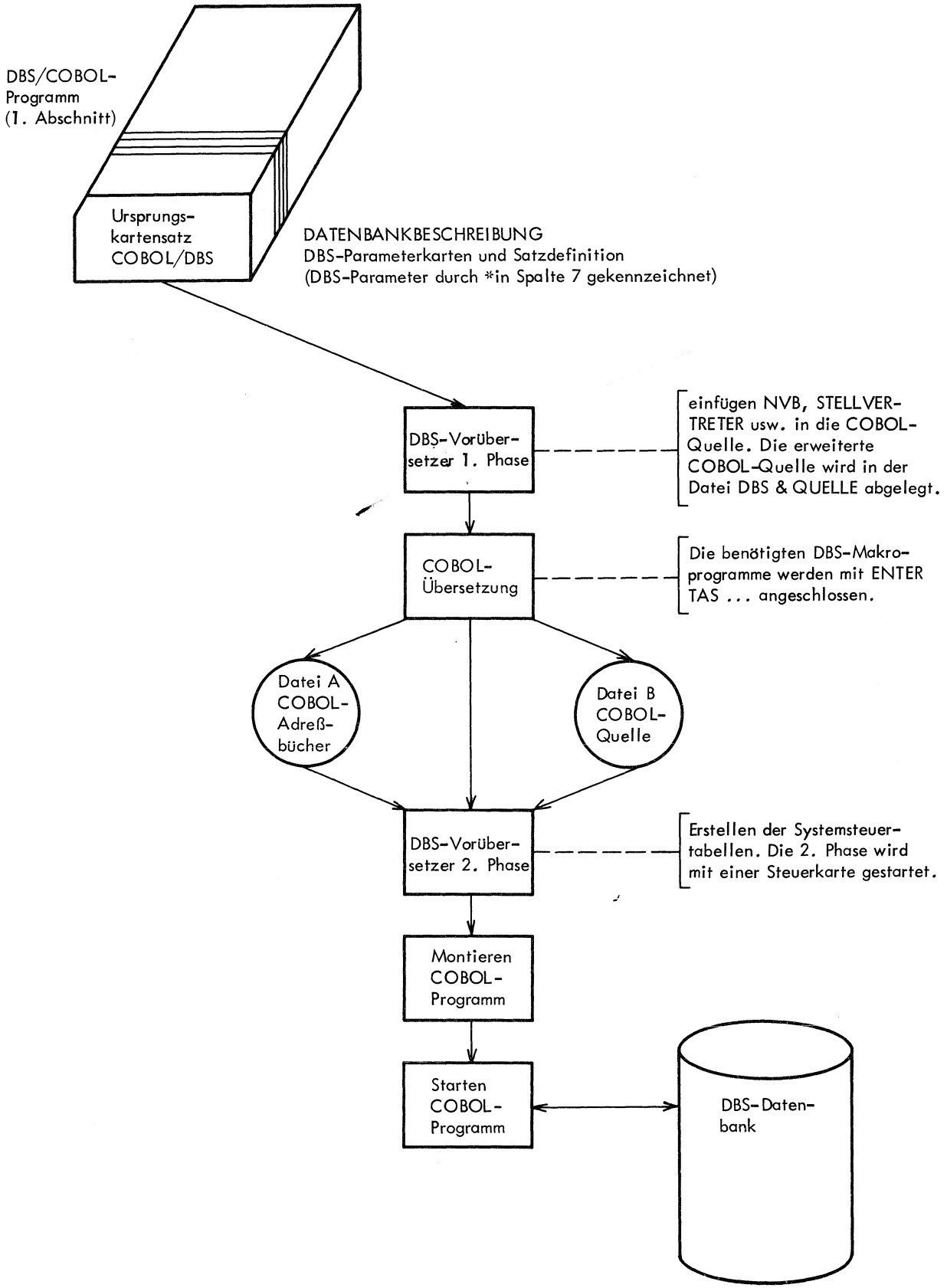


Bild 5.1 Systemzusammenhang COBOL/DBS

Wurde der Systembereich durch die Angabe SYSTEM = TEST auf Binärkarten gestanzt, so kann man sich in einem späteren Lauf das nochmalige Starten des DBS-Übersetzers ersparen durch Einfügen der Binärkarten anstelle der Startkarte für den DBS-Übersetzer (der Parameter SYSTEM = TEST muß aus dem Kartensatz nicht entfernt werden). Das Kartenpaket hat das Aussehen wie in Bild 6.2.

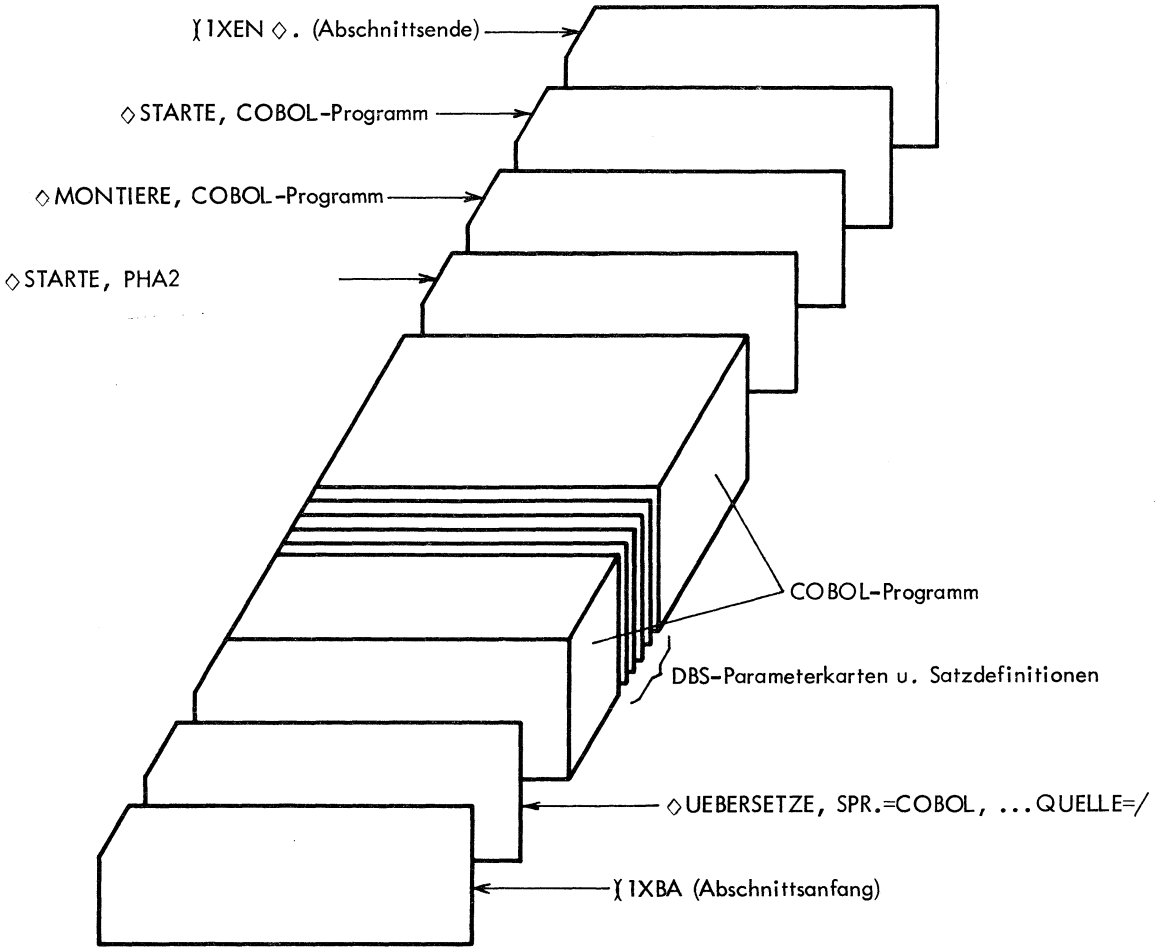


Bild 6.1 Kartenstapel für BS 3

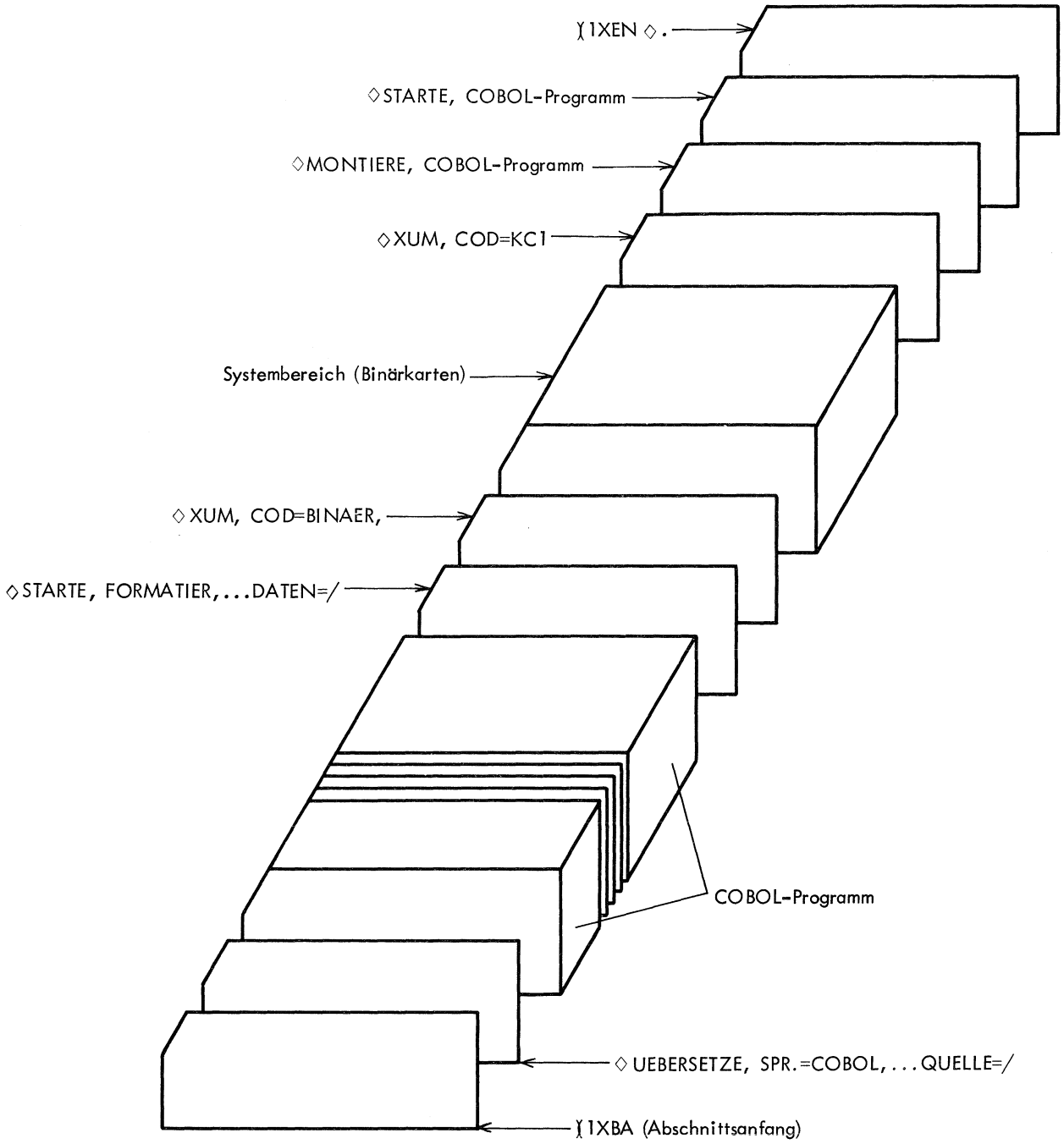


Bild 6.2 Eingabe des Systembereichs über Binärkarten Starten des Formatierers

Ist das COBOL-Programm fehlerfrei und will man sich für das weitere Arbeiten das Einlesen und Übersetzen der auf Lochkarten vorliegenden COBOL-Quelle ersparen, so besteht die Möglichkeit, das COBOL-Montageobjekt in die langfristige Datenhaltung (LFD) zu übernehmen.

Dies erfolgt durch die beiden Steuerkarten:

◇ LFDATEI, NAME = LFD-datei,
◇ BINAERAUS, MO = Name COBOL-Mont.objekt,
GERAET = LFD-datei

Die beiden Karten folgen unmittelbar auf die COBOL-Quellkarten, in jedem Fall stehen sie vor dem Montiere-Kommando für das COBOL-Programm. (Siehe auch TR 440-Kommando-Handbuch)

Will man sich die Montage ebenfalls ersparen, so ist bei dem BINAERAUS-Kommando der Programmname der COBOL-Quelle anzugeben und die BINAERAUS-Steuerkarte muß hinter der Montiere-Steuerkarte liegen.

In Bild 6.3 ist der DBS-Systembereich auf Binärkarten vorhanden und das Montageobjekt der COBOL-Quelle ist in der LFD abgelegt.

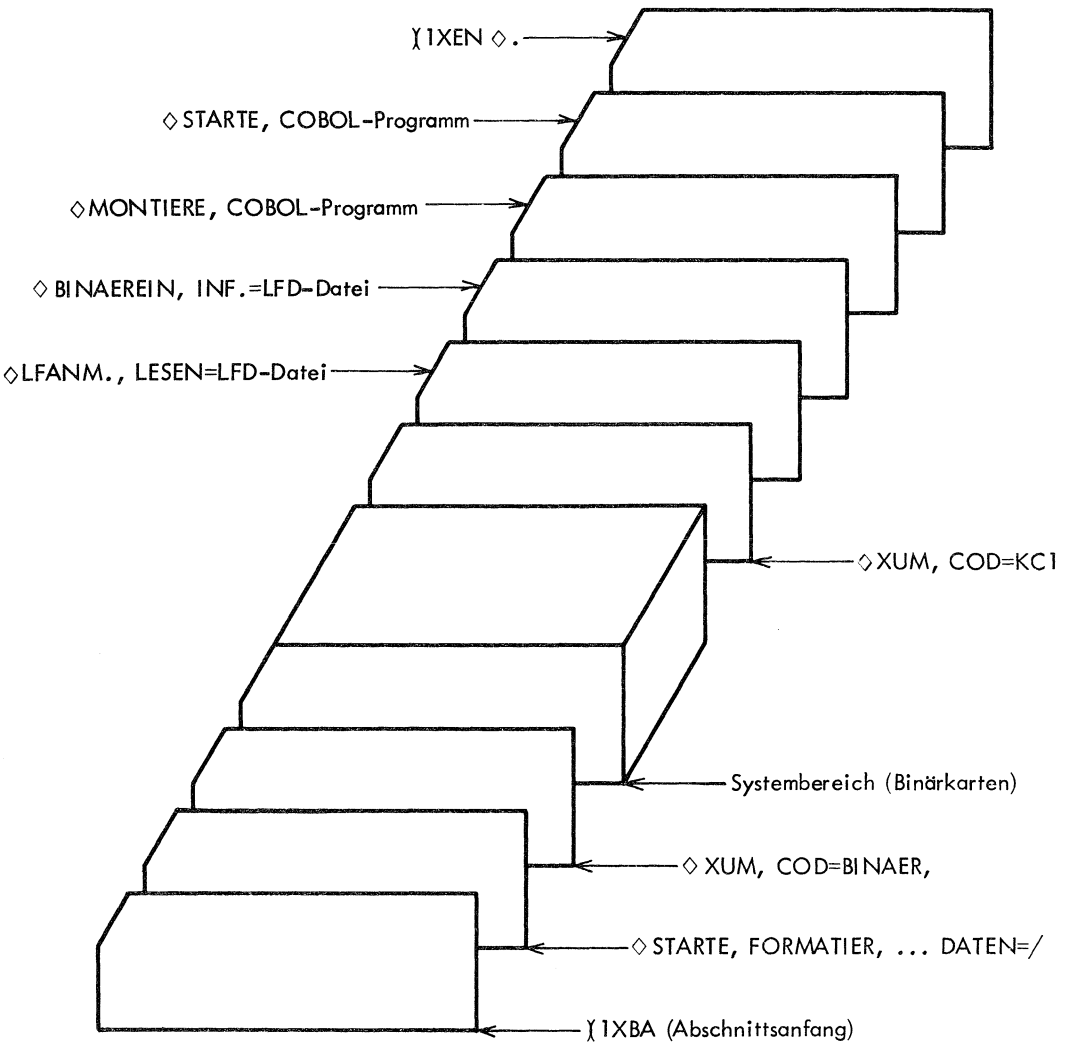


Bild 6.3 Das COBOL-Montageobjekt wird mit dem BINAEREIN-Kommando über eine LFD-Datei eingeschleust

7. FEHLERAUSGÄNGE

7.1. Fehlerbehandlung

Grundsätzlich werden zwei Arten von Fehlercodes unterschieden:

(1) Fehlercodes auf der Betriebssystemebene

(2) Fehlercodes des DBS

zu (1):

Detallierte Informationen über diese Fehlerbehandlungen sind in dem Handbuch "TR 440 Systemdienste BS3" beschrieben.

zu (2)

Die Fehlerorganisation des DBS wird unterteilt in:

(2.1) Fehlercodes der Datenmanipulationssprache (DMS-Fehlercodeliste)
(2.2) Fehlercodes des Input/Output-Controllers (IOC-Fehlercodeliste)
(2.3) Fehlercodes der Datenbank-Utilities (DBU-Fehlercodeliste)

Durch eine funktionelle Fehlercodeinterpretation ist eine eindeutige und schnelle Fehlerlokalisierung gewährleistet. Bei der Programmierung muß nur auf folgendes geachtet werden:

Nach jedem abgesetzten DBS-Befehl muß unbedingt das Feld FEHLERCODE abgefragt werden.

Bei FEHLERCODE = 0, wurde vom DMS die gewünschte Leistung erbracht;

bei FEHLERCODE > 0 gelten folgende Modi:

(2.1) 0 < FEHLERCODE ≦ 30
Der Fehler ist in der Datenmanipulationssprache aufgetreten. Der Benutzer kann nach entsprechender Fehlerbehandlung in seinem Anwendungsprogramm fortfahren (siehe DMS-Fehlercodeliste).

(2.2) 31 ≦ FEHLERCODE ≦ 580
Der Fehler ist im Input/Output Controller aufgetreten. DBS entzieht dem Benutzer die Regie, erstellt automatisch Systemdumps und schließt das Benutzerprogramm ab.

(2.3) 581 ≦ FEHLERCODE ≦ 700
Der Fehler ist in einer der Datenbank-Utilities aufgetreten. Fehlerbehandlung siehe entsprechende DBU-Fehlercodeliste.

7.2. Fehlercodes für die DMS

1 Satzname nicht gefunden
2 Kettenname nicht gefunden
3 Kettentabelle unbesetzt oder unvollständig besetzt (Anker- oder Vorgängeradresse fehlen)
4 Kette weder vorgänger- noch ankerverarbeitbar
5 Satz weder indexiert noch random bzw. bei HOLSTV nicht indexiert gespeichert
6 Nachfolger nicht vorhanden
8 Satz nicht gefunden
9 Feldname nicht in zu änderndem Satz definiert
10 Vorgänger ist Anker
11 Satz ist Duplikat
12 kein Platz im Indexbereich
13 Ankersatz nicht gefunden
15 Direktadresse (NVB) nicht zulässig
16 kein Platz im Datenbereich
17 kein zulässiger Schlüssel
18 DBS-Datei ungültig
19 Dateiname ungültig
20 maximale Pufferzahl überschritten
21 Bereichsende erreicht
22 Vor HOLIND kein HOLSTV abgelaufen
23 Es wurde kein Satz dieses Typs gespeichert oder geholt oder aber er ist gelöscht worden
25 Seitennummer unzulässig
26 Satz mit Ordnungsbegriff schon vorhanden
27 Kettenname gehört nicht zum aktuellen Satz oder aber der umzukettende Satz wurde nicht geholt

7.3. Fehlercodes im Input/Output-Controller

31 Schreibfehler
32 Lesefehler
33 Adresse nicht in der Puffergewichtstabelle
34 SN kleiner DATBGN
35 SN größer DATEND
36 SN kleiner BGNSL
37 SN größer INDEND
38 LN nicht belegt
39 LN belegt
40 Stellvertretersatz nicht gefunden
41 Schlüssel nicht in Indexseite
42 Kein Platz in der Seite (KIS < LNG)
43 Satz nicht in der Seite (STEADR ≧ FSAADR)
44 Der GDS gehört nicht in die Kettenkette des 'AKTADS'

45 SN kleiner INDBGN
46 SN größer BGNSL
47 AKTSDS nicht in der TYPTAB
48 TYP (LTW) nicht in der TYPTAB
49 Fehler beim Lesen eines speicheroptimierten Satzes
50 Fehler beim Zurückschreiben eines speicheroptimierten Satzes
51 Fehler beim Zurückschreiben eines speicheroptimierten Satzes, wenn der Datenteil ausgelagert werden muß
52 unbesetzt
53 unbesetzt
54 unbesetzt
55 unbesetzt
56 unbesetzt
57 unbesetzt
58 unbesetzt
59 unbesetzt
60 unbesetzt
61 unbesetzt
62 unbesetzt
63 unbesetzt
64 unbesetzt
65 unbesetzt
66 unbesetzt
67 unbesetzt
68 unbesetzt
69 unbesetzt
70 Primärdaten
71 Einzufügende Daten
72 unbesetzt
73 unbesetzt
74 unbesetzt
75 unbesetzt
76 unbesetzt
77 unbesetzt
78 unbesetzt
79 unbesetzt
80 unbesetzt

Den angeführten Fehlercodes wird noch eine weitere Ziffer vorangesetzt, je nach dem, in welchem Makro der Fehler festgestellt wird.

0 = PUST
1 = QHOL
2 = QLAD
3 = QLOE
4 = QSPE
5 = QWAF

Beispiele:

31 Schreibfehler: in PUST bei direktem Aufruf
131 Schreibfehler: in PUST und zwar in QHOL
439 LN nicht belegt: in QSPE
570 Primärdaten: in QWAF

7.4. Fehlercodes der DBS-Utilities

Die Fehlermeldungen sind aus der Beschreibung der entsprechenden DBS-Utilities zu entnehmen.

FEHLERCODE / DMS MODUL	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
ABSCHL																														
AENFLD									*							*	*						*			*				
AENKTN		*	*										*		*		*						*				*			
AENSTZ	*										*					*	*						*			*				
HOLAKT	*																						*							
HOLANK		*	*	*																										
HOLDAT																					*				*					
HOLDIR															*															
HOLEN	*				*			*									*													
HOLIND						*																*								
HOLKET		*																												
HOLNAC		*				*																								
HOLSTV	*				*			*									*													
HOLVOR		*	*							*																				
HOVOIN						*																*								
LOESCH	*																						*							
OEFFNE																		*	*	*										
SPEICH	*										*	*	*			*	*									*				

Bild 7.1 Fehlermatrix der DMS

In den nachfolgenden Tabellen werden die möglichen Fehlerursachen der verschiedenen DMS-Befehle aufgelistet.

DMS-Befehl	Fehler-code	MÖGLICHE FEHLERURSACHEN
AENFLD	9	AENFLD bezieht sich immer auf den zuletzt bearbeiteten Satz. Zumindest einer der nach USING aufgeführten Feldname gehört nicht zu dem zuletzt bearbeiteten Satz.
	11	Ein Satz mit dem Schlüssel der in dem zu ändernden Satz eingetragen werden soll, ist bereits gespeichert. Da Schlüsselduplikate nicht erlaubt sind, muß entweder -sofern möglich- der bereits gespeicherte Satz gelöscht werden, oder aber für den zu ändernden Satz ein noch nicht vergebener Schlüssel definiert werden.
	16	Für einen Satz für den DATENVERDICHTUNG definiert wurde, ergibt sich nach der Änderung eine neue Satzlänge die im Bereich nicht mehr untergebracht werden kann. Da es sich in der Regel um eine ganz geringe Fehlkapazität handeln wird, empfiehlt es sich den Bereich auf zu löschende Sätze zu durchsuchen und somit Kapazität für den geänderten Satz zu schaffen.
	17	Es wurde als Schlüssel SPACE oder HIGH VALUE angegeben. Diese Eintragung sind für den IOC reserviert und dürfen nicht vom Benutzer als Schlüssel vergeben werden.
	23	Der zu ändernde Satz ist nicht der zuletzt bearbeitete.
	26	Es wurde versucht, ein Sortierfeld für eine sortierte Kette, für die Duplikate verboten definiert wurde, zu ändern. Dabei wurde eine in dieser Kette bereits vorhandener Ordnungsbegriff angegeben.
AENKTN		
	2	Nach USING wurde kein gültiger DBS-Kettenname angegeben.
	3	Der alte Ankersatz kann von DBS nicht gefunden werden, da die Kette weder Vorgänger noch ankerverarbeitbar definiert wurde und der Benutzer diese Adresse dem DBS nicht mitgeteilt hat. Dieser Fehler wird behoben wenn der zu ändernde Gliedsatz über den Anker bereitgestellt wird.
	13	Für den neuen Ankersatz wurde ein Schlüssel eines bisher noch nicht gespeicherten Satzes angegeben.
	15	Die angegebene Direktadresse adressiert a) nicht den Datenbereich b) nicht den Bereich des Ankersatzes c) zwar den Bereich des Ankersatzes aber nicht den Ankersatz. Die gewünschte DIREKTADRESSE wird wie folgt angegeben Seitennummer x 64 + Liniennummer
	17	Der für den neuen Anker angegebene Schlüssel ist ungültig (SPACE oder HIGH VALUE).
	23	Der zu ändernde Satz wurde nicht geholt.
	27	Der hinter USING angegebene Kettenname gehört nicht dem aktuellen Satz oder aber der umzukettende Satz wurde nicht geholt.
AENSTZ		
	1	Hinter USING wurde kein gültiger Satzname angegeben
	11	wie bei AENFLD
	16	
	17	
	23	
	26	

DMS-Befehl	Fehler-code	MÖGLICHE FEHLERURSACHEN
HOLAKT	1	Hinter USING wurde kein gültiger Satzname angegeben.
	23	Es wurde noch kein Satz dieses Typs gespeichert bzw. geholt oder aber er ist gelöscht worden.
HOLANK	2	Hinter USING wurde kein gültiger Kettenname angegeben
	3	Die Kettentabelle ist unbesetzt es wurde noch kein Satz dieser Kette (weder Anker- noch Gliedsatz) geholt.
	4	Für die angegebene Kette ist nur die Adresse des aktuellen Gliedsatzes u.U. auch die Adresse eines Nachfolgers bekannt, da die Kette weder Anker noch vorgängervearbeitbar ist, kann der Anker nicht geholt werden.
HOLDAT	21	Es werden alle Sätze zwischen SN_ANFANG und SN-ENDE liegenden Seiten ausgegeben. Waren zwischen SN-ANFANG und SN-ENDE liegenden Seiten leer, so konnte natürlich kein Satz ausgegeben werden und es wird die Bearbeitung ebenfalls mit FEHLERCODE 21 beendet.
	25	Zumindest eine der in SN-ANFANG oder SN-ENDE angegebenen Seiten, gehört nicht mehr zum Datenbereich.
	15	Die in DIREKTADRESSE angegebene Adresse gehört nicht zum Datenbereich. Die gewünschte DIREKTADRESSE wird wie folgt angegeben: SN (Seitennummer) x 64 + Liniennummer).
HOLEN	1	Hinter USING wurde ein ungültiger Satzname angegeben.
	5	Mit HOLEN kann nur ein random oder indexiert sequentiell gespeicherter Satz geholt werden. Für den hinter USING angegebenen Satznamen wurde weder random noch indexiert sequentiell als Ablageform definiert.
	8	Es wurde noch kein Satz mit dem angegebenen Schlüssel gespeichert.
	17	Es wurde ein ungültiger Schlüssel (SPACE oder HIGH VALUE) angegeben.
HOLIND	6	Es gilt keinen Nachfolger im Indexbereich. Der zuletzt zur Verfügung gestellte Stellvertretersatz war bereits der Letzte.
	22	Vor HOLIND wurde kein HOLSTV aufgerufen. HOLIND wird durch HOLSTV versorgt. Es ist daher vor dem ersten Aufruf von HOLIND unbedingt HOLSTV aufgerufen.
HOLKET	2	Der hinter USING angegebene Kettenname ist ungültig.
HOLNAC	2	Der hinter USING angegebene Kettenname ist ungültig
	6	Es kann kein weiterer Nachfolger der angegebenen Kette bereitgestellt werden, das Kettenende wurde bereits erreicht.
HOLSTV	1	Hinter USING wurde ein ungültiger Satzname angegeben
	5	Für den hinter USING angegebenen Satznamen wurde nicht Ablage indexiert sequentiell definiert.
	8	Es wurde noch kein Satz mit dem angegebenen Schlüssel gespeichert und ein Satz mit einem höheren Schlüssel ist nicht vorhanden.
	17	Es wurde ein ungültiger Schlüssel (SPACE oder HIGH VALUE) angegeben.
HOLVOR	2	Hinter USING wurde ein ungültiger Kettenname angegeben.
	3	Für die angegebene Kette wurde weder Anker- noch Vorgängerverkettung definiert oder aber es wurde noch kein Satz dieser Kette (weder Anker noch Gliedsatz) geholt.

DMS-Befehl	Fehler-code	MÖGLICHE FEHLERURSACHEN
	10	Der zuletzt bereitgestellte Satz ist der Anker der Kette.
HOVOIN	6	Es kann kein Vorgänger im Indexbereich geholt werden. Der zuletzt zur Verfügung gestellte Stellvertretersatz war der log. 1. Stellvertretersatz.
	22	Vor HOVOIN wurde kein HOLSTV aufgerufen. HOVOIN wird durch HOLSTV versorgt. Es ist daher vor dem ersten Aufruf vom HOVOIN unbedingt HOLSTV aufzurufen.
LOESCH	1	Hinter USING wurde ein ungültiger Satzname angegeben
	23	Der zu löschende Satz ist nicht der aktuelle Satz der Datenbank.
OEFFNE	18	Die Datenbank wurde nach der letzten Bearbeitung nicht durch ABSCHL abgeschlossen und befindet sich somit in einem undefinierten Zustand. Es ist der letzte Lauf (selbstverständlich mit ABSCHL) zu wiederholen, um die Datenbank in einem definierten Zustand zu versetzen.
	19	Hier können folgende Fehler a) Hinter USING wurde nicht NVB als Versorgung angegeben b) Im NVB steht nicht der Datenbankname c) Die Datenbank wurde nicht vor Aufruf von OEFFNE eingeschleust.
	20	Die in PUFFERZAHL (NVB) angegebene Pufferzahl ist größer 100.
SPEICH	1	Hinter USING wurde ein ungültiger Satzname angegeben
	11	Es wurde bereits ein Satz mit dem gleichen Schlüssel des aktuellen Satzes gespeichert.
	12	Für einen indexiert sequentiell zu speichernden Satz ist kein Platz mehr im Indexbereich. Es wurde die in der DBS-Datenbeschreibung angegebene Anzahl Sätze überschritten
	16	Der zu diesem Satz gehörende Bereich ist belegt und kann den aktuellen Satz nicht mehr aufnehmen.
	17	Es wurde ein ungültiger Schlüssel (SPACE oder HIGH VALUE) angegeben.
	26	Es wurde versucht in eine sortierte Kette, für die Duplikate verboten definiert wurde, einen Satz mit einem in dieser Kette bereits vorhandenen Ordnungsbegriff zu speichern.

8.1. Der Aufbau der Dialogsprache

Mit der Datenbank-Dialogsprache steht dem Benutzer ein wirkungsvolles Mittel zum Abfragen und Verwalten von Datenbanken zur Verfügung. Die einzelnen Anweisungen (Dialogbefehle) der Dialogsprache ermöglichen je nach Aufgabenstellung gezielte Anfragen an eine Datenbank und/oder eine gezielte Änderung derselben.

Die Datenbank-Dialogsprache (DBS-Dialog) kann sich auf jede beliebige Datenbank aufsetzen und erfüllt die Funktionen eines Data-Management-Systems (Verwaltung der Datenbestände) und eines Informationssystems (beliebige Fragestellungen an das System, wobei die Ausführung der einzelnen Anweisungen von komplexen Bedingungen abhängig gemacht werden kann).

Das Dialogsystem besteht aus einem Dialogprozessor und einem Datenbankprozessor. Als Datenbankprozessor wird DBS 440 eingesetzt, es ermöglicht Aufbau und Verwaltung von integrierten Datenbeständen.

Der Sprachumfang des Dialogsystems setzt sich aus einer Anzahl von Schlüsselworten zusammen, mit denen der Benutzer seine "Anfragen", hier als Dialogprozeduren bezeichnet, formulieren kann (Näheres siehe Kap. 4).

Unter einer Dialogprozedur wird eine Folge von Dialog-Anweisungen verstanden, die mit einer SUCHEN-Anweisung beginnt und durch eine ENDE-Anweisung abgeschlossen wird.

Bei der Eingabe einer Dialogprozedur können diese Schlüsselworte beliebig verkürzt werden, solange Eindeutigkeit gewährleistet ist. Bei der Dialoganweisung AUSGEBEN z.B. werden auch die verkürzten Eingaben AUSGEB, AUSG oder AUS verstanden.

Die Ausführung der einzelnen Dialoganweisungen werden von sog. Prozessor-Prozeduren übernommen, im folgenden kurz als Moduln bezeichnet. Für jede Dialoganweisung existiert ein solcher Modul.

Der Dialogprozessor läßt sich in folgende funktionelle Einheiten gliedern:

a) Zentrale Steuerung

Die Zentrale Steuerung (RAHMEN) ist der zentrale Programmteil innerhalb des Dialogprozessors. Seine wesentlichen Aufgaben sind:

- Annahme der eingegebenen Anfragen
- Weiterleitung der Ausgaben an die Konsole
- Aufruf des Dialogentschlüsslers
- Aufruf der benötigten Moduln

b) Dialogentschlüssler

Nach der Eingabe einer Dialogprozedur wird vom RAHMEN der Dialogentschlüssler aufgerufen.

Durch den Dialogentschlüssler werden die eingegebenen Anweisungen übersetzt und für die benötigten Moduln des Dialogprozessors wird ein entsprechender Versorgungsblock aufgebaut.

Der Entschlüssler unterzieht eine eingegebene Dialogprozedur einer Syntaxprüfung. Syntaktisch fehlerhafte Dialogprozeduren werden nicht gestartet.

Es kann auf Wunsch ein entsprechendes Fehlerprotokoll auf der Konsole und im Ablaufprotokoll ausgegeben werden.

c) Versorgungsblock (VBL)

Der Versorgungsblock stellt eine Abbildung einer übersetzten Dialogprozedur dar. In ihm sind alle Angaben enthalten, die zur Ausführung einer Dialoganweisung und der damit verbundenen Versorgung der entsprechenden DBS-Befehle benötigt werden.

Aufgrund der Eintragungen im VBL ruft der RAHMEN die notwendigen Moduln auf.

d) Moduln (Prozedurteile)

Für jede Dialoganweisung gibt es einen Modul. Aufruf der Prozeduren erfolgt durch den RAHMEN aufgrund der Eintragungen im VBL.

Die Prozeduren liegen bereits in startfähiger Form vor. Sie stellen den eigentlichen Kontakt mit dem Datenbankprozessor her durch die entsprechenden DBS-Befehle. Die dafür erforderlichen Argumente (Parameter) werden dem VBL entnommen.

Der Vorteil dieser Strategie liegt darin, daß bei der Entschlüsselung der Dialogprozedur keine C o d e g e n e r i e r u n g vorgenommen werden muß, wie etwa bei kompilativen Dialogsystemen (GIS).

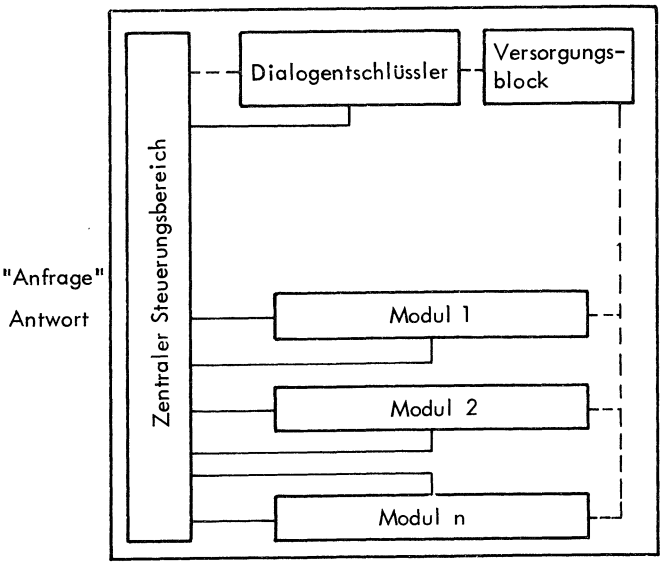


Abb: Dialogprozessor

8.2. Erstellen der DBS-Datenbank

Das Erstellen der DBS-Datenbank erfolgt durch COBOL/DBS-Programme. Die für den DBS-Prozessor erforderlichen Systemsteuertabellen werden nach der Übersetzung des COBOL-Programms aufgrund von DBS-Parametern aufgebaut und bilden einen Teil der Datenbank.

Soll für eine DBS-Datenbank Dialogverkehr möglich sein, so wird dies durch einen entsprechenden DBS-Parameter (*DIALOG =]) beim Aufbau der Datenbank angegeben. Dadurch wird vom DBS-Vorübersetzer ein erweiterter Systemsteuerbereich angelegt, der die für einen Dialogverkehr notwendigen Steuertabellen (Feldtabellen) enthält.

Eröffnen des Dialogs:

Will ein Benutzer mit der Datenbank im Dialog verkehren, muß er den Dialogprozessor durch ein STARTE-Kommando vom Terminal aus starten. Nachdem sich der Dialogprozessor mit einer Ausgabe gemeldet hat, sind Datenbankname und Paßwort einzugeben:

```
□ STARTE, DIALOG, L1□.
* DBS-DIALOG
* DATENBANKNAME□: DBS-DATEN□.
* PASSWORT□: PASS□ .
* EINGABE□:
```

Mit der Eingabe einer Dialogprozedur kann nun begonnen werden.

Nach Eingabe des Datenbanknamens wird vom Dialogprozessor durch Aufruf des DBS-Befehles OEFFNEN der Systemsteuerbereich in den Kernspeicher (KSP) geladen. Über die Paßwortangabe wird geprüft, ob der Zugriff auf die Datenbank erlaubt ist.
Bei Eingabe eines falschen Datenbanknamens oder eines falschen Paßwortes wird eine der folgenden Fehlermeldungen ausgegeben:

- DATENBANK NICHT VORHANDEN
- PASSWORT FALSCH
- ZUGRIFF NICHT ERLAUBT

Im Dialogprozessor gibt es nur eine Satzzone der Maximallänge einer DBS-Seite. Sämtliche Satzzonen der DBS-Datenbank werden auf diese Satzzone abgebildet, also immer in denselben Bereich gelesen. Weitere Sätze werden durch den entsprechenden DBS-Befehl automatisch nachgeladen.

Diese Satzzone, genauso wie Nachrichtenvermittlungsblock (NVB) und STELLVERTRETER, sind fest im Dialogprozessor einassembliert.

Nach Ablauf von OEFFNEN trägt der Dialogprozessor in die operatorlaufspezifische Tabelle (TYPTAB) für alle Satzzonen der DBS-Datenbank die operatorlaufrelative Adresse seiner eigenen Satzzone ein.

8.3. Systemsteuerbereich

Durch Angabe des Parameters "*DIALOG =]" wird der normale Systemsteuerbereich (Stapelverarbeitung) erweitert durch Steuertabellen (Feldtabellen), die für einen Dialogverkehr nötig sind.

Für jeden in der Datenbank definierten Satztyp wird eine Feldtabelle angelegt, in der sämtliche Feldnamen des Satztyps entsprechend der Reihenfolge der Felder im Satz stehen, sowie die zugehörigen Feldattribute (Feldlänge, Feldcharakter, evtl. Masken für formatisierte Ausgaben usw.). Weiterhin ist vermerkt, ob ein Feld noch in weitere bezeichnete Unterfelder gegliedert ist.

8.4. Allgemeine Form einer Dialogprozedur

Dem Benutzer stehen zahlreiche Anweisungen zur Verfügung, mit deren Hilfe er eine Anfrage an die Datenbank formulieren kann. Diese Anweisungen lassen sich aufgrund ihrer Funktionen in folgende Gruppen einteilen:

- Ausgabe von Sätzen oder Feldinhalten
- Ändern von Feldinhalten
- Rechenoperationen mit Feldinhalten
- Informieren über Ketten- bzw. Satzstrukturen
- Ausgabe von Listen

Die allgemeinste Form einer Anfrage hat folgendes Aussehen:

```
SUCHEN <DBS-Bereich>;
[ WENN <Bedingung>;]
Op1;
Op2;
.
.
.
Opn;
ENDE;
```

Eine Dialogprozedur beginnt mit der Anweisung SUCHEN, um anzugeben, mit welchem DBS-Bereich gearbeitet werden soll. Es können dann beliebig viele Dialoganweisungen erfolgen. Den Abschluß bildet die ENDE-Anweisung.

Die Operationen Op_1 bis Op_n stellen die eigentlichen Anweisungen der Dialogsprache dar.
Die Anweisung ENDE kennzeichnet das Ende einer Suchanfrage. Sie entfällt, wenn nach der Operation Op_n ein weiteres SUCHEN gegeben wird.

Soll die Verarbeitung eines Bereiches von Bedingungen abhängig gemacht werden, muß die WENN-Klausel benutzt werden; diese muß unmittelbar der SUCHE-Anweisung folgen.
Will ein Benutzer z.B. mit einem durch Satznamen bezeichneten Satz arbeiten (im Bereich sequentiell abgelegt), so wird dies durch folgende Suchanweisung erreicht:

```
SUCHEN SATZ = <Satzname>;
```

Die Eingabebereitschaft für eine Dialogprozedur wird angezeigt durch die Meldung:

* EINGABE␣:

Ist eine Dialogprozedur fehlerfrei und kann sie ausgeführt werden, so folgt unmittelbar darauf die Ausgabe der Ergebnisse.
Sind Fehler vorhanden, so wird dies durch die Meldung "*FEHLERAUSG. ENTSCHLUESSLER" angezeigt.
Darauf folgt die Ausgabe der fehlerhaften Prozedur mit Ausgabe des Fehlerortes (durch| und folgende ///) und der Fehlerart.

Nach Beenden der Prozedur, angezeigt durch die Ausgabe "*ENDE PROZEDUR", oder nach Ausgabe einer fehlerhaften Prozedur wird durch die Meldung "*WEITER J/N␣:" angefragt, ob eine neue Prozedur angegeben werden oder ob der Dialog abgebrochen werden soll.

Alle Ausgaben des Dialogprozessors beginnen mit einem *.

Näheres siehe Kapitel 8.6 .

8.5. Einfache Anfragen

Die Syntax der Dialogsprache erlaubt ohne besondere EDV-Kenntnisse relativ komplexe Anfragen an das System.

Beispiel 1:

```
SUCHEN KETTE = PERSONAL, SL = N31V4;
SUMME GEHALT;
ENDE;
```

Durch die obige Prozedur läßt sich sofort das Gehaltsvolumen der spezifizierten Datei (Satztyp) PERSONAL ermitteln, wenn man weiß, daß durch die Anweisung SUCHEN KETTE = PERSONAL und der Schlüsselangabe SL = N31V4 die PERSONAL-Datei ermittelt wird und durch die Anweisung SUMME GEHALT sämtliche Gehaltsfelder aufsummiert werden und das Ergebnis anschließend vom Dialogprozessor ausgegeben wird.

Soll das Gehaltsvolumen für eine bestimmte Tarifgruppe ermittelt werden, so läßt sich dies durch folgende Prozedur erreichen:

Beispiel 2:

```
SUCHEN KETTE = PERSONAL, SL = N31V4;
WENN TAGRU GL T3;
SUMME GEHALT;
ENDE;
```

Der Dialogprozessor gibt anschließend aus:

3 450 673,40

Eine verbesserte Ausgabe wird durch die Prozedur in Beispiel 3 erreicht:

```
SUCHEN KETTE = PERSONAL, SL = N31V4;
WENN TAGRU = T3;
SUMME GEHALT (R);
ENDE PERSONAL;
LISTE 'SUMME GEHALT TARIFGR. T3:' GEHALT;
ENDE;
```

Auf dem Terminal erscheint die Ausgabe:

SUMME GEHALT TARIFGR. T3: 3 450 673,40

Die Spezifikation R (≙Reserviere) hinter dem Feldnamen GEHALT verhindert, daß direkt nach Abarbeitung der Kette bei Erkennen des Pseudobefehls ENDE PERSONAL das Gehaltsvolumen ausgegeben wird. Der in einem Arbeitsfeld zwischengespeicherte Wert wird erst durch die Angabe des Feldnamens in der anschließenden Anweisung LISTE ausgegeben.

Aufbau eines Satzes der PERSONAL-Datei:

NAME	VORNAME	PERNU	KOSTEL	GEHALT	TARGU	STEUKLA

Abb.: 2

8.6. SUCHEN Lokalisierung eines Satzes oder einer Kette

Jede Dialogprozedur wird durch die Anweisung SUCHEN eingeleitet. Mit ihr legt der Benutzer fest, mit welchem Satztyp oder mit welcher Kette der DBS-Datenbank er arbeiten will.

```
                ⎧ KETTE = <Kettenname>  ⎫ [⎧ SL          ⎫ = <Wert>]
SUCHEN          ⎩ SATZNAME = <Satzname> ⎭ ⎩ <Feldname>  ⎭
                [KETTE = <Kettenname>,]   ⎧ ANKER      ⎫
                                          ⎨ VORGAENGER ⎬
                                          ⎩ NACHFOLGER ⎭
```

Das Schlüsselwort SUCHEN kann bei der Eingabe weggelassen werden. Auch die Angabe K = <Kettenname> und S = <Satzname> ist ebenso eindeutig und vereinfacht die Eingabe am Terminal.
Sind Kettenname und Satzname nicht definiert, so gibt der Dialogentschlüssler folgende Fehlermeldung:

KETTENNAME NICHT VORHANDEN

oder

SATZNAME NICHT VORHANDEN

Zum besseren Verständnis der internen Abläufe sind Blockdiagramme an gegebener Stelle eingefügt.

Folgende Fälle sind für SUCHEN zu unterscheiden:

1. SUCHEN K = <Kettenname>, SL = <Wert>;

Es sollen nur die Gliedersätze der angegebenen Kette bearbeitet werden. Der Ankersatz selbst soll nicht in die Verarbeitung mit einbezogen werden.
Der Dialogprozessor sucht zuerst den Anker dieser Kette und positioniert danach sofort auf den ersten Gliedsatz.
Es ergibt sich hieraus folgende Strategie:

Der Dialogprozessor interpretiert (liest) zuerst den Systemsteuerbereich bevor der DBS-Prozessor mit einem DBS-Befehl versorgt wird.

Durch die Angabe K = <Kettenname> kann im Systemsteuerbereich der Satzname des Ankers dieser Kette gefunden werden. Mit diesem Satznamen wird der DBS-Befehl "HOLEN <Satzname> SATZ" gegeben.

Der Schlüssel SL = <Wert> wurde vorher in die Satzzone des Dialogprozessors eingetragen.
Der DBS-Prozessor sucht nun den entsprechenden Satz mit den Angaben Satzname und Schlüssel und transportiert den Satz in den aktuellen Puffer und in die entsprechende Satzzone. Die Kettentabelle wird dabei aktualisiert.
Durch den internen Aufruf des DBS-Befehl "HOLEN NACHFOLGER IN <Kettenname> KETTE" können nun die einzelnen Gliedersätze der Kette gefunden werden.

Die Angabe SL = <Wert> setzt voraus, daß der Ankersatz ISQ oder RANDOM gespeichert ist. Dies wird vom Dialogentschlüssler abgeprüft.

2. SUCHEN K = <Kettenname>, <Feldname> = <Wert>;

Die Operation SUCHEN in dieser Form wird angegeben bei verketteten, sequentiell gespeicherten Sätzen.

Im Systemsteuerbereich wird abgeprüft, ob für den Satztyp eine sequentielle Speicherung vorliegt. (Diese Prüfung erfolgt bereits durch den Dialogentschlüssler.)
Wurde ABLAGE = SEQUENTIELL durch DBS-Parameter nicht definiert, so wird als Fehlermeldung ausgegeben:

KEIN SEQUENTIELLER SATZTYP.

Aus dem sequentiellen Bereich werden die Sätze intern mit dem DBS-Befehl "HOLDAT" nacheinander eingelesen, bis die Bedingung <Feldname> = <Wert> erfüllt ist.

Der gefundene Satz mit der Bedingung <Feldname> = <Wert> ist Anker der Kette <Kettenname>. Nachfolgend wird intern der DBS-Befehl "HOLEN NACHFOLGER IN <Kettenname> KETTE" abgesetzt.

3. SUCHEN S = <Satzname>, SL = <Wert>;

Mit dieser Suchanweisung soll nur ein einziger Satz angesprochen werden (Ablage ISQ oder RANDOM).
Mit dem DBS-Befehl "HOLEN <Satzname> SATZ" wird auf den bezeichneten Satz positioniert. Der durch die Angabe SL = <Wert> definierte Schlüssel wurde vorher in die Satzzone des Dialogprozessors eingetragen. Der Satz kann sowohl Anker als auch Glied einer Kette sein.

Bei Angabe dieser Suchanweisung ist keine Kettenbarbeitung möglich. (Von dem so lokalisierten Satz können, wenn er der Anker ist, mehrere Ketten ausgehen).

Werden anschließend Dialoganweisungen gegeben, die sich auf das Bearbeiten einer Kette beziehen, wie etwa die Operation SUMME, so führt dies zu Fehlern.

Im Laufe der Bearbeitung kann aber durch die Anweisung

SUCHEN K = <Kettenname>, { ANKER / VORGAENGER / NACHFOLGER } ;

auf eine bestimmte Kette positioniert werden, wenn die weitere Verarbeitung dies erfordert.

4. SUCHEN S = <Satzname>, <Feldname> = <Wert>;

Bearbeiten von sequentiell gespeicherten Sätzen.

Es wird geprüft, ob eine sequentielle Speicherungsform vorliegt. Aus dem sequentiellen Bereich werden die Sätze intern mit dem DBS-Befehl "HOLDAT" nacheinander eingelesen, bis die Bedingung <Feldname> = <Wert> erfüllt ist.

5. SUCHEN S = <Satzname>;

Diese Suchoperation wird gegeben bei sequentiell bzw. indexsequentiell (ISQ) gespeicherten Sätzen, wobei eine evtl. vorhandene Verkettung unberücksichtigt bleiben soll. Die Entscheidung sequentiell oder ISQ ergibt sich aus vorangegangener Definition.

a) sequentiell gespeicherte Sätze

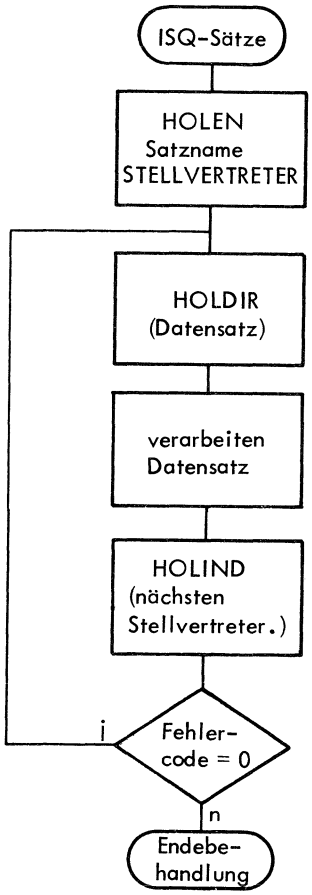
Durch die Angabe <Satzname> kann im Systembereich (SYSBER) geprüft werden, welche Speicherungsform vorliegt.

b) indexsequentiell gespeicherte Sätze

Es sollen ISQ-gespeicherte Sätze vom niedrigsten Schlüssel an aufsteigend bis zum höchsten Schlüssel verarbeitet werden. Dabei müssen auch die Stellvertretersätze sequentiell gelesen werden.

Der erste Satz der Folge von Stellvertretersätzen wird intern mit dem DBS-Befehl "HOLEN <Satzname> STELLVERTRETER" gelesen. Die Nachfolger dieses Satzes in der Stellvertreterliste werden intern durch den Befehl "HOLEN IN INDEXBEREICH" (HOLIND) gelesen.
Der eigentliche Datensatz kann nun mit dem DBS-Befehl "HOLDIR" (HOLEN DIREKT) gelesen werden.

In einem Blockdiagramm stellt sich dies, aus DBS-Sicht gesehen, folgendermaßen dar:



6. SUCHEN [K = <Kettenname>,] { ANKER / VORGAENGER / NACHFOLGER } ;

Diese Anweisung kann nur dann gegeben werden, wenn bereits durch ein vorangegangenes SUCHEN der unter Punkt 1 - 5 beschriebenen Form auf einen Satz innerhalb einer Kette positioniert wurde (Es muß aber in jedem Fall eine Verkettung vorliegen).

Beispiel:

SUCHEN K = <Kettenname>, SL = <Wert>;
Op1;
Op2;
SUCHEN K = <Kettenname>, ANKER;
Op3;
.
.
.
Opn;
ENDE;

Die Operation dient hier dazu, um von einem beliebigen Gliedsatz auf den durch Kettenname definierten Ankersatz zu kommen.

Kettenname in der Suchanweisung SUCHEN K = <Kettenname>, ANKER kann auch eine andere Kette bezeichnen, als die, die zu diesem Gliedsatz geführt hat. Wird der Anker der gleichen Kette verlangt, so kann K = <Kettenname> weggelassen werden. Die Kette muß in diesem Fall nicht ankerverarbeitbar sein.

Ist der Kettenname angegeben, so wird geprüft, ob er mit dem in der vorangegangenen Suchanweisung gegebenen Kettennamen übereinstimmt.

Sind die Kettennamen verschieden, so bedeutet dies, daß auf eine neue Kette positioniert werden soll. In diesem Fall muß der Typ des Gliedsatzes ankerverarbeitbar sein.

Durch die Angabe VORGAENGER bzw. NACHFOLGER kann zu einem positionierten Satz der Vorgänger- oder Nachfolgersatz geholt werden.

Anmerkung:

In der 2. Ausbaustufe des Dialogprozessors können bei den Angaben SUCHEN K = <Kettenname>, <Feldname> = <Wert> bzw. SUCHEN S = <Satzname>, <Feldname> = <Wert>; auch ISQ-Sätze verarbeitet werden (allerdings keine RANDOM-Sätze).

Bei ISQ-Sätzen wird mit dem DBS-Befehl HOLSTV, nachdem 0 als Schlüssel in die Satzzone eingetragen wurde, der niedrigste Stellvertretersatz gelesen. Danach wird mit dem DBS-Befehl "HOLDIR" der zugehörige Datensatz gelesen und geprüft, ob er die Bedingung <Feldname> = <Wert> erfüllt. Durch abwechselndes Absetzen der DBS-Befehle "HOLEN NACHFOLGER IN STELLVERTRETER" und "HOLDIR" können die ISQ-Sätze sequentiell gelesen werden.

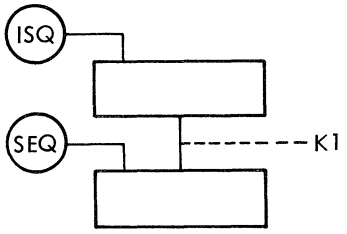
Beispiel eines Protokolles:

```
*EINGABE¤:
SUCHEN KETTE=ORTSKETTE,SL=KONSTANZ;
 AUSGEBEN NAME,VORNAME,STRASSE,PERSNR1;
ENDE;¤.

 NAME : VOWINKEL
 VORNAME : LORE
 STRASSE : IRGENDWO
 PERSNR1 : 999999

 NAME : NEUMANN
 VORNAME : JOSEPH
 STRASSE : BODAN
 PERSNR1 : 777777

 NAME : KOZDERA
 VORNAME : FRANZ
 STRASSE : WIEHNER-HOF
 PERSNR1 : 414141

 NAME : ROST
 VORNAME : HORST
 STRASSE : ALPENBLICK
 PERSNR1 : 707070
*ENDE PROZEDUR
```

8.6.1. Komplexe Kettenstrukturen

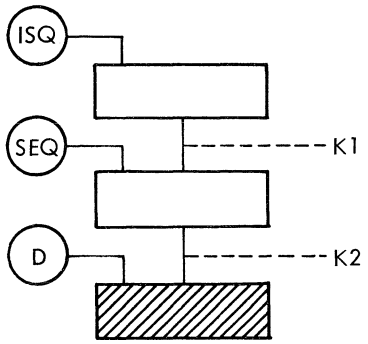
Im folgenden werden einige Suchanfragen für komplexe Kettenstrukturen beschrieben, ausgehend von der unter Punkt 1 beschriebenen einfachsten DBS-Kettenstruktur:

1. Die Ankersätze sind ISQ gespeichert und die Gliedersätze sequentiell (SEQ).



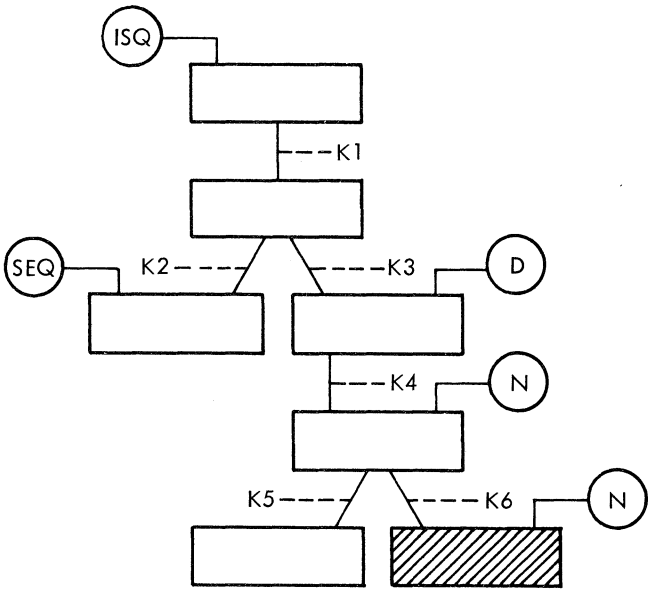
SUCHEN K = K1, SL = S1;
Op1;
.
.
.
Opn;
ENDE;

2. Bearbeitung der Sätze in der Kette K2:



SUCHEN K = K1, SL = S1;
SUCHEN K = K2, FELD = F1;
Op_1;
.
.
.
Op_n;
ENDE;

3. Bearbeitung der Sätze in der Kette K6:



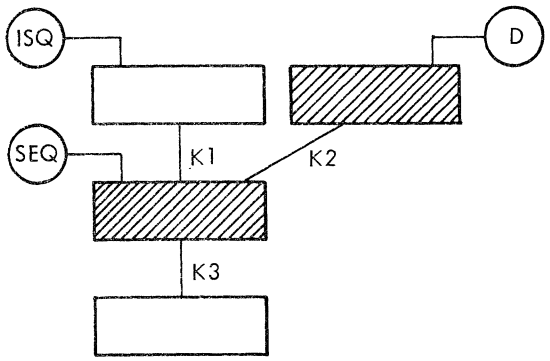
SUCHEN K = K1, SL = S1;
SUCHEN K = K3, FELD1 = F3;
SUCHEN K = K4, FELD2 = F4;
SUCHEN K = K6, FELD3 = F6;
Op_1;
.
.
.
Op_n;
ENDE;

Die vorher erwähnte Anfrage in weniger schreibaufwendiger Form:

K = K1; SL = S1;
K = K3; FELD1 = F3;
K = K4; FELD2 = F4;
K = K6; FELD3 = F6;
Op_1;
.
.
.
Op_n;
ENDE;

(Feld1, FELD2, FELD3 sind Feldnamen innerhalb des jeweiligen Satztyps; F3, F4, F6 sind die Inhalte dieser Felder.)

4. Nach Bearbeiten einer Kette K1 liegt die Notwendigkeit vor, eine Kette 2 zu bearbeiten. Kette 2 kann nur verarbeitet werden, wenn für sie die Verkettungsform "mit ANKER" angegeben ist.



SUCHEN K = K1, SL = S1;
Op_1;
.
.
Op_i;
} Bearbeitung der Kette K1

SUCHEN K = K2, ANKER; } Übergang auf Kette K2

Op_{i+1};
.
.
Op_n;
ENDE;

8.7. WENN-Klausel

Die Ausführung von Anweisungen kann von Bedingungen abhängig gemacht werden. Durch die WENN-Klausel ist die Möglichkeit gegeben, Bedingungen zu stellen.

WENN <Bedingung> [{ UND / ODER } <Bedingung>]^n ;

<Bedingung> : = <Feldname> V <Wert> [,<Wert>]^n;

V : = <Vergleichsoperator>

<Vergleichsoperator> : = { ZWISCHEN | GLEICH | GROESSER | KLEINER | {GROESSEER GLEICH / GG} | {KLEINER GLEICH / KG} }

<Wert> : = { <alphanumerischer String> | <Festkommazahl> | <Gleitkommazahl> | <Bitvariable> }

Für den Vergleichsoperator ist folgende Darstellung gleichbedeutend, sofern sich diese Zeichen am Terminal darstellen lassen:

<Vergleichsoperator> : = { = | > | < | ≥ | ≤ }

Beispiele:

1. Wenn Tarifgruppe T1 oder T2 oder T3:

   WENN TAGRU = T1, T2, T3;

   umständlicher formuliert:

   WENN TAGRU = T1 ODER TAGRU = T2 ODER TAGRU = T3;

2. Wenn Tarifgruppe T1 oder T2 oder T3 und der Beruf = Programmierer:

   WENN TAGRU = T1, T2, T3 UND BERUF = PROGR;

3. Wenn Gehalt größer 3500 und Beruf = Systemspezialist (SS):

   WENN GEHALT> 3500 UND BERUF = SS;

4. Wenn Gehalt größer 3000 und Beruf = Fachberater (FB) oder Systemspezialist (SS):

   WENN GEHALT> 3000 UND BERUF = FB, SS;

Sollen auf den gleichen Satztyp verschiedene WENN-Klauseln zur Anwendung kommen, so wird dies durch folgende Anfrage erreicht:

SUCHEN KETTE = <Kettenname>, SL =<Wert>;
WENN <Bedingung-1>;
Op_1;
Op_2;
.
.
.
Op_i;
WENN <Bedingung-2>;
Op_{i+1};
.
.
.
Op_n;
ENDE

Die WENN-Klausel veranlaßt den Dialogprozessor, jeweils wieder auf den Anfang der Kette zu positionieren. Abhängig von der entsprechenden Bedingung werden dann die nachfolgenden Anweisungen ausgeführt.

Beispiel eines Protokolles:

```
*EINGABE¤:
SUCHEN K=ORTSKETTE,SL=KONSTANZ;
 WENN NAME=BALIG,BECKER,FROEHLICH
 ODER VORNAME=HELMUT,OTTO,MATTHIAS;
 AUSGEBEN NAME,VORNAME,STRASSE;
END;¤.

 NAME : BECKER
 VORNAME : GUSTAV
 STRASSE : KREUZL.STR. 27

 NAME : BALIG
 VORNAME : LUDWIG
 STRASSE : RUPPANERSTR.22

 NAME : FROEHLICH
 VORNAME : BRIGITTE
 STRASSE : VEILCHENALLEE

 NAME : FROMM
 VORNAME : HELMUT
 STRASSE : MARKGRAFENSTR.3

 NAME : GUEGEL-FRANK
 VORNAME : OTTO
 STRASSE : ALTER WALL 2

 NAME : GUMBEL
 VORNAME : MATTHIAS
 STRASSE : BRANDESTR. 35
*ENDE PROZEDUR
```

8.8. Die Anweisungen der Dialogsprache

Im folgenden werden die einzelnen Befehle (statements) der Dialogsprache näher beschrieben.

Es kann zusätzlich noch eine Spezifikation (R) angegeben werden. Diese hat nur Bedeutung im Zusammenhang mit der Anweisung LISTE und ist nur erlaubt in den Anweisungen SUMME, ZSUM, ZAEHLEN und DURCHSCHNITT.

8.8.1. AENDERN Ändern von Feldinhalten

AENDERN <Wertzuweisung>;

<Wertzuweisung> : = <Feldname>=<Wert>[, <Feldname>=<Wert>]

Durch die Anweisung AENDERN können beliebige Feldinhalte innerhalb ein und desselben Satzes verändert werden. Diese Anweisung bezieht sich nur auf das Ändern von Feldern. Es können keine Sätze neu eingetragen oder vollständig überschrieben werden.

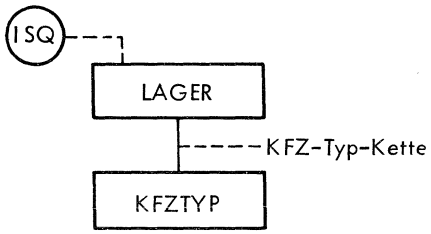
Mit AENDERN können einfache Felder, Sortierfelder und Schlüsselfelder geändert werden. AENDERN kann vom Inhalt bestimmter Felder des angesprochenen Satzes abhängig gemacht werden.

Beispiel 4:

In der Datei KFZ-Typen (Satztyp = KFZ-Typ) sollen für den KFZ-Typ VW1200 die technischen Daten geändert werden.

Die KFZ-Typen sind in einer KFZ-Typ-Kette miteinander verknüpft. Anker der Kette ist der Satz LAGER = WOLFSBURG (Satztyp ist LAGER).

Es liegt dabei folgende DBS-Kettenstruktur vor:



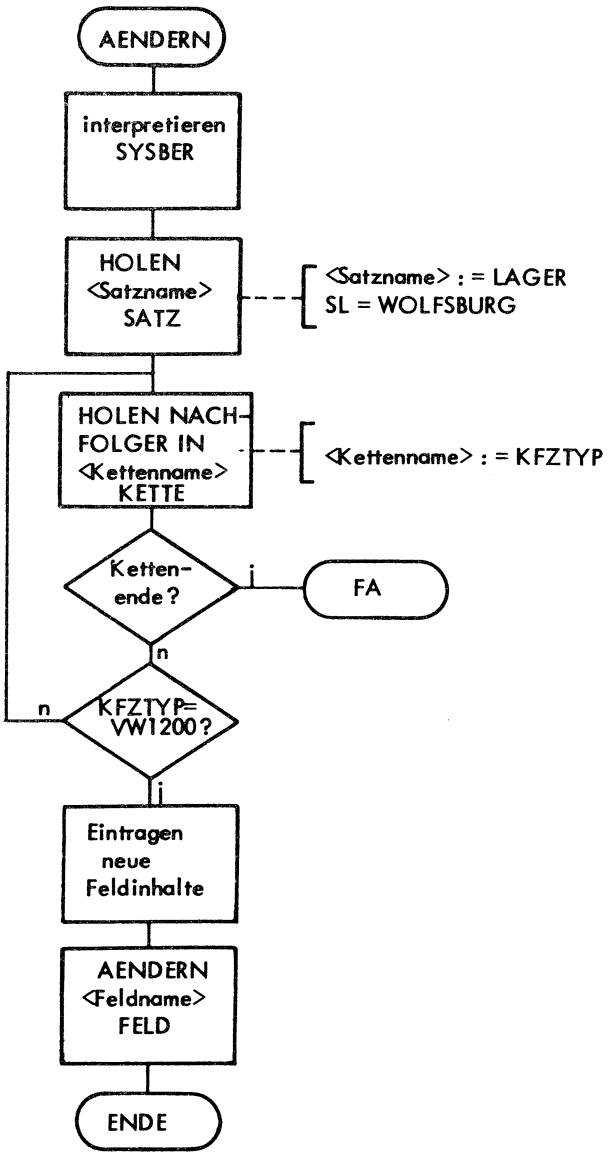
Ein Satz in der KFZ-Typ-Kette habe z.B. folgenden Aufbau:

KFZ-TYP	BAU-JAHR	DATEN				REIFEN	
		PS	LAENGE	BREITE	HOEHE	TYP	DRUCK

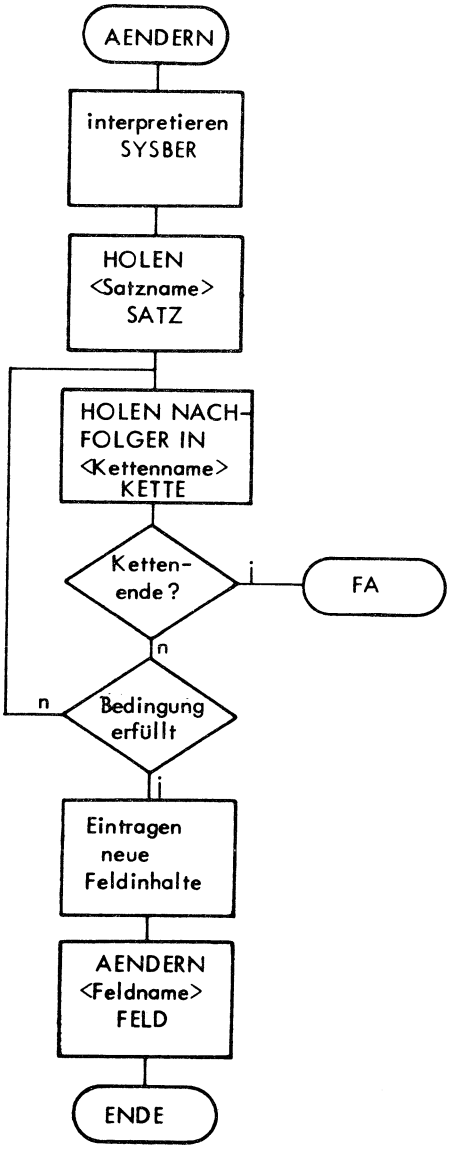
Abb. 3

```
SUCHEN KETTE = KFZTYP, SL = WOLFSBURG;
WENN KFZTYP GL VW1200;
AENDERN PS = 34,
LAENGE = 4070,
BREITE = 1550, HOEHE = 1500;
ENDE;
```

Der Dialogprozessor setzt intern folgende DBS-Befehle ab (Blockdiagramm):



Interner Ablauf der Anweisung AENDERN aus DBS-Sicht gesehen:



Beispiel eines Protokolles:

```
*EINGABE¤:
SUCH K=ORTSKETTE,SL=KONSTANZ;
 WENN NAME=FROEHLICH;
 AUSGEBEN NAME,VORNAME,STRASSE;
 AENDERN VORNAME=HEIDRUN;
 AUSGEBEN NAME,VORNAME,STRASSE;
END;¤.

 NAME : FROEHLICH
 VORNAME : SABINE
 STRASSE : GOTTLIEBERST.18

 NAME : FROEHLICH
 VORNAME : HEIDRUN
 STRASSE : GOTTLIEBERST.18
*ENDE PROZEDUR

*WEITER J/N¤:J¤.

*EINGABE¤:
SUCH K=ORTSKETTE,SL=KONSTANZ;
 WENN NAME=FROEHLICH;
 AUSG NAME,VORNAME,STRASSE;
 AENDERN VORNAME=BRIGITTE,STRASSE=VEILCHENALLEE;
 AUSG NAME,VORNAME,STRASSE;
END;¤.

 NAME : FROEHLICH
 VORNAME : HEIDRUN
 STRASSE : GOTTLIEBERST.18

 NAME : FROEHLICH
 VORNAME : BRIGITTE
 STRASSE : VEILCHENALLEE
*ENDE PROZEDUR
```

8.8.2. AUSGEBEN Ausgeben von Feldinhalten

AUSGEBEN { <Feldname> [,<Feldname>]^{n}[,ANZAHL = <Wert>]
SATZ

Beispiel 5:

```
SUCHEN K = KFZTYP, SL = WOLFSBURG;
AUSGEBEN KFZTYP;
ENDE;
```

Es werden sämtliche auf dem Lager WOLFSBURG abgelagerten KFZ-Typen ausgegeben.

Beispiel 6:

```
SUCHEN K = KFZTYP, SL = WOLFSBURG;
 WENN KFZTYP = VW1200;
   AUSGEBEN PS (R), LAENGE (R), BREITE (R);
   ENDE KFZTYP;
   LISTE 'LAGER = WOLFSBURG, KFZTYP = VW1200';
   LISTE 'PS:', PS, 'LAENGE:', LAENGE, 'BREITE:',
         BREITE;
ENDE;
```

Es erscheint folgende Ausgabe auf dem Bildschirm:

```
PS: 34
LAENGE: 4070
BREITE: 1550
```

Aufbau eines Satztyps KFZTYP:

KFZTYP	DATEN			
	PS	LAENGE	BREITE	HOEHE

Beispiel eines Protokolles:

```
*EINGABE¤:
SUCHEN KETTE=ORTSKETTE,SL=RADOLFZELL;
 AUSGEBEN NAME,VORNAME,STRASSE;
ENDE;¤.

 NAME : RAEUCHLE
 VORNAME : EBERHARD
 STRASSE : HEGELPLATZ

 NAME : BLANK
 VORNAME : KURT
 STRASSE : PFAFFSTRASSE
*ENDE PROZEDUR

*WEITER J/N¤:J¤.

*EINGABE¤:
SUCHEN KETTE=ORTSKETTE,SL=KONSTANZ;
 WENN VORNAME=EDITH,LORE,HELLA,BRIGITTE;
 AUSGEBEN NAME,VORNAME,STRASSE;
END;¤.

 NAME : VOWINKEL
 VORNAME : LORE
 STRASSE : IRGENDWO

 NAME : BERGMANN-ENGEL.
 VORNAME : HELLA
 STRASSE : SCHIFFSTR. 24

 NAME : FROEHLICH
 VORNAME : BRIGITTE
 STRASSE : VEILCHENALLEE
*ENDE PROZEDUR
```

8.8.3. SUMME Summenbildung

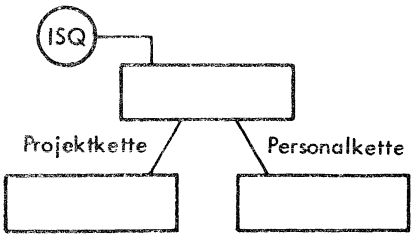
SUMME <Feldname> [,<Feldname>]^{n};

Die Anweisung SUMME organisiert einen Zugriff zu sämtlichen Sätzen einer lokalisierten Kette und summiert alle Werte eines angesprochenen Feldes in dieser Kette auf. Alle Sätze innerhalb der Kette müssen den gleichen Satztyp haben. Sind Sätze eines anderen Typs in der Kette, werden diese bei der Summenbildung ignoriert.

Sind die zu verarbeitenden Sätze sequentiell oder ISQ gespeichert, so werden sie in aufsteigender Folge gelesen und das entsprechende Feld aufsummiert.

Es können auch mehrere Felder innerhalb eines Satzes angegeben werden, die in einer positionierten Kette aufsummiert werden sollen.

Dem nachstehenden Beispiel liegt folgende DBS-Kettenstruktur zugrunde:



Beispiel 7:

SUCHEN K = PERSONAL, SL = TC;
SUMME GEHALT;
ENDE;

Im obigen Beispiel wird das Gehaltsvolumen der PERSONAL-Kette gebildet (Mitarbeiter der Abteilung TC). Durch die Anweisung SUMME wird Satz für Satz der angesprochenen Kette geholt, der entsprechende Feldinhalt aufsummiert und zwischengespeichert.

Bei Erreichen des Ketten-Ende wird anschließend das Ergebnis ausgegeben.

Durch die WENN-Klausel kann das Summenbilden von bestimmten Feldinhalten abhängig gemacht werden. Es soll beispielsweise die Summe der Gehaltsfelder nur dann gebildet werden, wenn in dem Feld TAGRU (Tarifgruppe) der Wert T3 steht:

Beispiel 8:

SUCHEN K = PERSONAL , SL = N31V4;
WENN TAGRU = T3;
SUMME GEHALT;
ENDE;

Nach Ablauf der Prozedur wird ausgegeben:

50 420,40

Eine ausführliche Ausgabe wird erreicht durch die folgende Anfrage:

Beispiel 9:

SUCHEN K = PERSONAL, SL = N31V4;
WENN TAGRU = T3;
SUMME GEHALT (R);
ENDE PERSONAL;
LISTE 'SUMME GEHALT TAGRU T3:', GEHALT;
ENDE;

Es wird anschließend ausgegeben:

SUMME GEHALT TAGRU T3: 50 420,40

Das in Klammern eingeschlossene R bei der Anweisung SUMME besagt, daß das SUMME-Ergebnis reserviert wird und erst bei expliziter Angabe in der Anweisung LISTE mit ausgegeben wird.

Beispiel 10:

Das Gehaltsvolumen soll bestimmt werden für die Tarifgruppe T3 und Beruf PROGRAMMIERER.

SUCHEN K = PERSONAL, SL = N31V4;
WENN TAGRU = T3 UND BERUF = PROGR;
SUMME GEHALT;
ENDE;

Beispiel 11:

Es soll das Gehaltsvolumen für die Mitarbeiter bestimmt werden, die Programmierer sind und einer der Tarifgruppen T2, T3 oder T4 angehören:

SUCHEN K = PERSONAL, SL = N31V4;
WENN TAGRU = T2, T3, T4 UND BERUF = PROGR.;
SUMME GEHALT;
ENDE;

8.8.4. ZSUM Bilden von Zwischensummen

ZSUM <Feldname>;

Syntax der WENN-Klausel für die Operation ZSUM:

WENN <Feldname>;

Die Operation ZSUM ermöglicht das Bilden von Zwischensummen in Abhängigkeit des Gruppenwechsels eines bestimmten Feldes.

Beispiel 12:

```
SUCHEN K = PERSONAL, SL = N31V4;
WENN TAGRU;
ZSUM GEHALT;
ENDE;
```

Es werden alle Zwischensummen der Gehaltsfelder bei gleicher Tarifgruppe gebildet. Nach Abarbeitung der Kette erscheint folgende Ausgabe auf dem Terminal:

```
TAGRU T3:   105 084 300,40
TAGRU T4:    88 405 500,20
TAGRU T5:     6 430 450,10
```

(Bei der Ausgabe wird automatisch der Feldname mit angegeben.)

8.8.5. ZAEHLEN Zielgruppenanfrage

ZAEHLEN [<Arbeitsfeldname>];

<Arbeitsfeldname> : = Name eines Feldes, das in einer nachfolgenden LISTE-Anweisung angesprochen werden kann.

Die Anweisung ZAEHLEN ermöglicht die Feststellung einer Anzahl von Sätzen in Abhängigkeit vom Inhalt, sie ist deshalb nur sinnvoll unter Verwendung einer vorangehenden WENN-Klausel.

Die Anweisung ZAEHLEN kann z.B. bei folgender Fragestellung angewendet werden:
Wieviele Einwohner gibt es in der Stadt XYZ, die männlich (M) und unter 21 Jahre alt sind.

Beispiel 13:

```
SUCHEN K = EINWOHNER, SL = XYZ;
WENN SEX = M UND ALTER < 21;
ZAEHLEN;
ENDE;
```

Hinter der Anweisung ZAEHLEN kann ein Name zur Bezeichnung eines Arbeitsspeicherfeldes stehen, das in einer nachfolgenden LISTE-Anweisung angesprochen werden kann.

Beispiel 14:

```
SUCHEN K = EINWOHNER, SL = XYZ;
WENN SEX = M UND ALTER < 21;
ZAEHLEN Z1 (R);
ENDE EINWOHNER;
LISTE 'ANZAHL EINW., MAENNL. UNTER 21:', Z1;
ENDE;
```

Ausgabe z.B.:

```
ANZAHL EINW., MAENNL. UNTER 21: 2701
```

8.8.6. RECHNEN Rechenoperationen

RECHNEN {<Feldname> / <Arbeitsfeldname>} = {<Feldname> / <Wert>} A {<Feldname> / <Wert>}

A : = <arithmetischer Operator> : = {ADD / SUB / MUL / DIV} ≙ {+ / - / * / /}

Mit RECHNEN können beliebige Rechenoperationen ausgeführt werden. Dabei können verschiedene Felder eines Satzes in die arithmetischen Operationen mit einbezogen werden. Die Ergebnisse werden in die Zielfelder eingespeichert und auf dem Terminal ausgegeben.

```
SUCHEN K = PERSONAL, SL N31V4;
WENN NAME = MEYER UND KOSTEL = 651;
AUSGEBEN GEHALT;
RECHNEN GEHALT = GEHALT + 250;
ENDE;
```

Auf dem Terminal erscheinen die Werte

```
1 450
1 700
```

8.8.7. DURCHSCHNITT Berechnen des Durchschnitts

DURCHSCHNITT <Feldname> [,<Feldname>]^{n};

Durch die Anweisung DURCHSCHNITT werden von den angegebenen Feldern die arithmetischen Durchschnittswerte gebildet und anschließend ausgegeben.

```
SUCHEN K = PERSONAL, SL N31V4;
WENN TAGRU = T3;
DURCHSCHNITTE GEHALT;
ENDE;
```

8.8.8. INFORMIEREN Informieren über Ketten- und Satzstruktur

Der Benutzer kann sich mit dieser Satzstruktur ausgeben lassen:

- wie oft ein bestimmter Satztyp verkettet ist,
- wie dieser Satztyp MASTER oder GLIED einer Kette ist
- die ANKERWAHL, wenn der betreffende Satztyp GLIED der Kette ist
- die ABLAGE, d.h. ob der Satz DIREKT, SEQUENTIELL, ISQ, NAHE oder RANDOM gespeichert ist.

INFORMIEREN <Satzname>

Die Angabe sieht dann folgendermaßen aus:

SATZTYP: <Satztyp-Nr.>

{ NICHT VERKETTET
| VERKETTET: <Kettenname> { GLIED / ANKER } }

[ANKERWAHL { AKTUELL
| MIT { SCHLUESSEL / <Feldname> } }]

ABLAGE: { DIREKT
| SEQ
| ISQ
| NAHE <Kettenname>
| RANDOM }

8.8.9. LISTE Textausgaben

LISTE '<Text>'<Feldname> [,'<Text>'<Feldname>]^{n};

Durch LISTE können die vom Dialogprozessor veranlaßten Ausgaben kommentiert werden. Dies ist vor allem dann von Bedeutung, wenn Konsolprotokolle über längere Zeiträume hinweg als Belege dienen sollen.
LISTE ermöglicht somit formatisierte Ausgaben zur Erzeugung entsprechender Druckbilder.

Im folgenden Beispiel werden nach der Bearbeitung des jeweiligen Satztyps die speziellen Ausgabeanweisungen Op_{i+1} - Op_n ausgeführt.
Stehen diese Operationen vor dem Pseudobefehl ENDE <Kettenname>, so werden sie sofort ausgeführt, wie die Anweisungen Op_1 - Op_i durchlaufen werden.

SUCHEN KETTE = <Kettenname>, SL = <Wert>;
WENN <Bedingung>;
Op_1;
Op_2;
.
.
.
Op_i;
ENDE <Kettenname>;
Op_{i+1};
.
.
.
Op_n
ENDE

8.8.10. ENDE

ENDE [{ <Satzname> / <Kettenname> }];

Durch die Operation ENDE, gefolgt von einem Semikolon, wird das Ende einer Dialogprozedur gekennzeichnet. Steht nach dieser Operation noch der Satz-/Kettenname der 1. zu dieser Dialogprozedur gehörenden SUCH-Operation, so bedeutet dies, daß nach der Operation ENDE noch LISTE-Befehle folgen, die nach Abarbeitung einer Kette oder eines Datenbereiches (ein einziges Mal) ausgeführt werden sollen. Nach dem so spezifizierten ENDE-Befehl dürfen nur noch LISTE-Befehle folgen.

8.8.11. Übersicht über die Syntax der Dialogsprache

1. Lokalisieren einer Kette oder eines Datenbereichs

SUCHEN { { KETTE = <Kettenname> / SATZNAME = <Satzname> } [, { SL / <Feldname> } = <Wert>] / [KETTE = <Kettenname>,] { ANKER / VORGAENGER / NACHFOLGER } } ;

2. WENN-Klausel

WENN <Bedingung> [{ UND / ODER } <Bedingung>]^{n} ;

<Bedingung> = : <Feldname> V <Wert> [, <Wert>]^{n}

= : <Vergleichsoperator>

<Vergleichsoperator> = : { ZWISCHEN / GLEICH / GROESSER / KLEINER / { GROESSER GLEICH / GG } / { KLEINER GLEICH / KG } }

<Wert> = : { <alphanumerischer String> / <Festkommazahl> / <Gleitkommazahl> / <Bitvariable> }

3. AENDERN

AENDERN <Wertzuweisung>

<Wertzuweisung> : = <Feldname> = <Wert> [, <Feldname> = <Wert>]^{n}

4. AUSGEBEN

AUSGEBEN { <Feldname> [, <Feldname>]^{n} [, ANZAHL = <Wert>] / SATZ }

5. SUMME

SUMME <Feldname> [, <Feldname>]^{n} ;

6. ZSUM

ZSUM <Feldname> ;

7. ZAEHLEN

ZAEHLEN [<Arbeitsfeldname>] ;

8. RECHNEN

RECHNEN { <Feldname> / <Arbeitsfeldname> } = { <Feldname> / <Wert> } A { <Feldname> / <Wert> } ;

A : = <arithmetischer Operator>

9. DURCHSCHNITT

DURCHSCHNITT <Feldname> [, <Feldname>]^{n} ;

10. INFORMIEREN

INFORMIEREN <Satzname> ;

11. LISTE

LISTE '<Text>'<Feldname> [,'<Text>'<Feldname>]^{n} ;

12. ENDE

ENDE $\left\{\begin{array}{l}\text{<Kettenname>}\\\text{<Satzname>}\end{array}\right\}$;

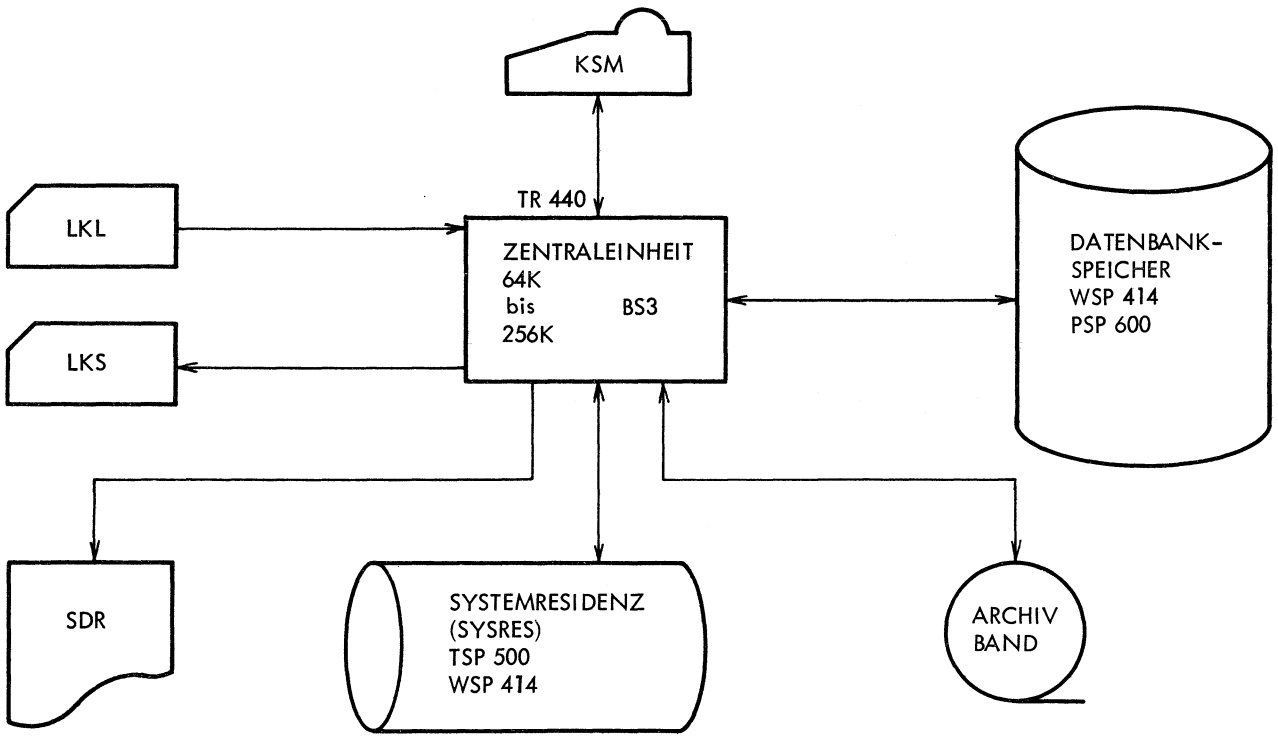
8.8.12. Fehlermeldungen

Im folgenden sind die z.Zt. möglichen Fehlermeldungen aufgelistet, eine Erläuterung wird später folgen.

- SCHLUESSELWORT FALSCH
- NICHT ERLAUBTE SUCH-OP
- ABBRUCH, ZU VIELE FEHLER
- NICHT ERLAUBTES ZEICHEN
- UNZULAESSIGE ZEICHENFOLGE
- KOMMA FALSCH
- MEHR ALS 1 BINDESTRICH
- FELDNAME NICHT VORHANDEN
- FELDNAME NICHT ERLAUBT
- MAX. ANZAHL FELDNAMEN UEBERSCHRITTEN
- KETTENNAME NICHT VORHANDEN
- SATZNAME NICHT VORHANDEN
- KEIN SEQUENTIELLER SATZTYP
- KEINE PROZ. VORHANDEN

9. ANHANG

9.1. Maschinenausstattung



9.2. DBS-Utilities

Dem Benutzer stehen eine Reihe von Utilities zur Verfügung, die durch ihre Dienstleistungen dazu beitragen, einen wirtschaftlichen Betrieb zu gewährleisten.

Im einzelnen sind dies:

* DBS-DUMP
* DBS-KOMPRESSOR
* DBS-STATISTIK
* DBS-REORGANISATION
* DBS-REGENERATION
* DBS-SICHERN
* DBS-LOGBUCH

In der augenblicklichen DBS-Version sind die Utilities:

* DBS-DUMP
* DBS-KOMPRESSOR

in das Gesamtsystem implementiert; an der Realisierung der o.g. Utilities wird im Augenblick gearbeitet.
Für die Handhabung dieser Utilities stehen speziell Anwendungsschriften zur Verfügung.

9.2.1. DBS-DUMP

Die Routine DBS-DUMP diagnostiziert auf Grund vorgegebener Parameter des Gesamtinhalt oder Teilbereiche der Datenbank auf dem Schnelldrucker.

Anwendung

Das Programm liegt als zuladbare Service-Routine in der LfD vor. Die ausführliche Handhabung kann in der Anwendungsbeschreibung nachgelegt werden.

Parameter

Der Benutzer hat zwei Möglichkeiten, die DBS-DUMP-Routine zu versorgen.

Durch Angabe von:

1. ⟨Seitenbeginn⟩, ⟨Seitenende⟩, A,
   werden innerhalb dieses Bereiches die Inhalte aller Sätze diagnostiziert.
   Beispiel:
   100, 800, A,

2. ⟨Seitenbeginn⟩, ⟨Seitenende⟩, ⟨Satztyp$_1$⟩, ⟨Satztyp$_2$⟩...
   werden innerhalb dieses Bereiches die Inhalte der durch Satztyp$_1$, Satztyp$_2$... spezifizierten Sätze diagnostiziert.
   Beispiel:
   100, 800, 3, 5, 21,
   Innerhalb der Seitenbereichsangabe 100 bis 800 werden die Inhalte sämtlicher Sätze definiert, die vom Satztyp 3, 5 oder 21 sind.

9.2.2. Testparameter

Die Testparameter werden am Ende des Installationsteils der Datenbankbeschreibung, also unmittelbar vor dem Gebietsteil eingetragen.

Die Reihenfolge ist:

* DATENBANKNAME = Datenbankname
* PASSWORT = Datenname
* SYSTEM = { NEU [RESERVIERE [QUELLE]] / TEST },
* DRUCK = { JA / NEIN / NUR PARAMETER }
* Gebiet.

usw.

SYSTEM

Aufgabe:

Angabe, ob eine neue Datenbank, oder nur eine Syntaxprüfung der DBS-Parameter vorgenommen werden soll.

Aufbau:

[* SYSTEM = { TEST / NEU [RESERVIERE [QUELLE]] }]

Erklärungen:

Es bedeutet SYSTEM =

1. NEU: Nach Erstellung der DBS-Steuertabellen wird die Datenbank formatiert und, wenn angegeben entspr. Reservierungen vorgenommen.

2. TEST: Die System-Steuertabellen werden binär ausgestanzt. Standardvorbesetzung ist SYSTEM = NEU

DRUCK

Funktion

Angabe, ob und wie die DBS-Parameter während der Testarbeiten ausgedrückt werden.

Format

```
[ *   DRUCK = { JA           } . ]
             { NEIN         }
             { NUR PARAMETER }
```

Erklärungen:

1. Der Parameter darf überall stehen und kann mehrmals vorkommen. Er gibt die gewünschte Protokollierung der DBS-Parameter an.

   Standardvorbesetzung ist DRUCK = JA

2. DRUCK = JA: Alle Karten zwischen

   * DATENBANKBESCHREIBUNG und
   * DBS-ENDE werden ausgedruckt.

   DRUCK = NEIN: Nur fehlerhafte Karten werden gedruckt.

   DRUCK = NUR PARAMETER: Es werden nur die DBS-Parameter (mit * in Spalte 7) ausgedruckt.

9.3. Glossary

Adresse
Die Adresse dient zum Wiederauffinden eines Speicherplatzes, sie kann absolut oder relativ zu einem festen Bezugspunkt angegeben sein.

Logische Adressen
werden bei Randomspeichern verwendet. Der Platz eines gespeicherten Satzes wird durch Seitennummer und Liniennummer innerhalb der Seite angegeben.

Anker
Als Anker wird der erste Satz in einer Kette bezeichnet. Da er als Einsprungspunkt in eine Kette dient, soll er weitgehend Daten enthalten, die sich auf alle Glieder der zugehörigen Kette beziehen.

Basissprache
Unter der Basissprache wird die Höhere Programmiersprache verstanden, die in einem Datenbanksystem den verfahrensorientierten Rahmen abgibt, um eine Verarbeitung der Daten im Arbeitsspeicher zu ermöglichen.

Block
Ein Satz bzw. eine Gruppe von aufeinanderfolgenden Sätzen, die als physische Einheit behandelt werden und die kleinste adressierbare Einheit auf einem peripheren Speicher darstellt. Ein Block wird auch als physischer Satz oder Datenträgersatz bezeichnet.

Blocken, Blockungsfaktor, Ladefaktor
Das Zusammenfassen mehrerer logischer Sätze in einem Block nennt man Blocken. Die Anzahl der Sätze, die zu einem Block zusammengefaßt werden können, ist aus dem Blockungsfaktor oder Ladefaktor ablesbar.

Datei
Eine Zusammenfassung logisch zusammengehöriger Sätze gleichen Typs.

Daten
Allgemeine Bezeichnung für Buchstaben, Ziffern, Symbole, Kombinationen von diesen bzw. Bezeichnung von Gegenständen, Begriffen, Situationen, Bedingungen und anderen Faktoren.

Datenbank
Eine Zusammenfassung von Dateien bzw. Datenbeständen, die die Basis für ein formalisiertes Informationssystem in einem abgeschlossenen Organisationsbereich abgibt und die wesentlichste Grundlage für die Informationswiedergewinnung (information retrieval) bildet.

Die Dateien, aus denen eine Datenbank aufgebaut ist, können nach der Struktur der in ihnen vorhandenen Datensätze unterschieden werden. Die Sätze einer Datei heißen formatiert, wenn sie bezüglich ihres Formates (Feldaufbau, Feldfolge innerhalb eines Satzes und Länge des betreffenden Satzes) eindeutig definiert sind. Andernfalls heißen sie unformatiert.

Datenbankprozessor
All jene Makros, die gewisse Dienstleistungen in einem Datenbanksystem erbringen. Der Datenbankprozessor ist zusammengesetzt aus der Datenmanipulationssprache (DMS) und dem I/O-Controller.

Datenbeschreibungssprache (DBB)
Die Datenbeschreibungssprache stellt für den Benutzer eine softwaremäßige Unterstützung zur Erstellung eines Schemas (Datendefinitionstabelle) des gesamten Datenbestandes dar.

Datenelement
Die kleinste logische Einheit, die sich nicht aus weiteren logischen Einheiten zusammensetzt.

Datenmanipulationssprache DMS
Die Datenmanipulationssprache stellt in operierenden Systemen eine Sprache dar, mit deren Hilfe der Programmierer eine Transaktion der Daten zwischen Benutzerprogramm und Datenbank durchführt.

Datenverdichtung
Unter Datenverdichtung versteht man das Reduzieren eines Datenbestandes auf seinen für die Verarbeitung wesentlichen Inhalt, also das Entfernen redundanter Informationen. Diese Verdichtung wird einerseits aus speichertechnischen Gründen angestrebt - um auf Großraumspeichern Platz zu sparen -, andererseits zur besseren und konzentrierteren Darstellung von Information bei der Auswertung.

Glied
Anker und Glieder sind Kettenelemente.

Hierarchie der Organisationseinheiten
Bit (bit), Zeichen (character), Datenelement (data element), Satz (record), Datei (file), Datenbestand (data base), Datenbank (data bank),

Indizierungsmethode
Ein Zugriff zu einem bekannten Datenelement aufgrund eines vorgegebenen Ordnungskriteriums setzt jeweils das Begriffspaar

Ordnungskriterium - Adresse

voraus.

Wird diese Verbindung mit einem Tabellenmechanismus (Indexliste) erzeugt, so spricht man von einer Indizierung.

Information
Sie ist eine logisch abgeschlossene Einheit, die sich aus Daten gemäß formaler Regeln zusammensetzt und von einem Sender (Mensch, Rechner), zu einem Empfänger (Mensch, Rechner) fließt. Unterscheidung in

unformatierte Informationen
formatierte Informationen

Bei unformatierten Informationen bestehen die Datenelemente aus den Worten der Umgangssprache, die Regeln ihrer Zusammensetzung aus der grammatikalischen Syntax.

Bei formatierten Informationen sind die Datenelemente Werte. Die Zusammensetzungsregeln ergeben sich aus vordefinierten Schemata; die Position der Information innerhalb eines Datensatzes bestimmt den Informationsinhalt.

Informationssystem
Informationssysteme sind Datenorganisationsformen auf Rechenanlagen, die bestimmte Vorgänge und Abläufe im Kommunikationsprozeß unterstützen. Diese Vorgänge und Abläufe müssen, damit sie überhaupt maschinell bearbeitet werden können, formalisier- oder automatisierbar sein.

Kette
Eine Kette ist eine Gruppe von Sätzen mit folgenden Eigenschaften

Jede Kette hat stets genau einen Anfang, den sogenannten Kettenanker.

Jede Kette kann beliebig viele Glieder haben.

Anker und Glieder heißen allgemein auch Kettenelemente. Im Anker steht die Adresse des ersten Kettengliedes, in diesem die Adresse des zweiten Gliedes usw., im letzten Kettenglied steht das Kettenendzeichen. DBS-Ketten sind sogenannte offene Ketten.

Maschinenabhängige Datenunterteilung
Bit (bit), Zeichen (character), Wort (word), Block (block), Datenträger (file),

Paßworte
Kennworte, die bestimmten Benutzern Zugang zu geschützten Bereichen bzw. Daten einer Datenbank ermöglichen. Mit Paßwörtern kann auch die Art des Zugriffs (Lesen, Schreiben, usw.) geregelt werden.

Regeneration
Um zu vermeiden, daß bedingt durch technische Pannen Datenbestände zerstört werden, müssen sogenannte Regenerationsverfahren eingesetzt werden. Der gesamte Datenbestand wird z.B. zusätzlich auf einem Magnetband geführt, das auch ständig aktualisiert sein sollte.

Reorganisation
In gewissen Zeitabständen werden Reorganisationsverfahren notwendig werden, die die zwischenzeitlich veränderten Datenbestände der Datenbank bezüglich Speicherplatz, Zugriffszeit etc. optimieren.

Seite
Das gesamte externe Speichermedium wird in Seiten gleicher Länge eingeteilt. Eine Seite kann aus einem oder mehreren Blöcken bestehen.

Speicherungs- und Verarbeitungsformen
Die Speicherungsform gibt an, wie der Bestand auf dem Datenträger organisiert und abgelegt ist.

Die Verarbeitungsform definiert die Art des Zugriffs auf diesen Bestand.

Utilities
Hierunter werden Dienstleistungsroutinen verstanden, die einen wirtschaftlichen Betrieb in einem Datenbanksystem gewährleisten. Beispiele sind u.a.:

DUMP, KOMPRESSOR, STATISTIK, REORGANISATION, REGENERATION, ARCHIVIERUNG

Zeichen
Eine Grundeinheit aus einem limitierten, definierten Vorrat von Symbolen, der für einen Kommunikationsbereich deklariert wurde.