KfK 2581
März 1978

DISCO: Eine Datenbankarchitektur für verteilte DV-Systeme auf der Basis logischer Dateien

E. Holler, H. Breitwieser, O. Drobnik,
C. Keil, U. Kersten, R. Knöpker, J. Krieger
Institut für Datenverarbeitung in der Technik

Kernforschungszentrum Karlsruhe

KERNFORSCHUNGSZENTRUM KARLSRUHE

Institut für Datenverarbeitung in der Technik

KfK 2581

DISCO: Eine Datenbankarchitektur für verteilte DV-Systeme auf der Basis logischer Dateien[+)]

E. Holler, H. Breitwieser, O. Drobnik, C. Keil, U. Kersten
R. Knöpker, J. Krieger

Kernforschungszentrum Karlsruhe G.m.b.H., Karlsruhe

[+)] Ergebnisbericht des mit Mitteln des Bundesministeriums für Forschung und Technologie geförderten Vorhabens "Datenbankarchitektur für verteilte DV-Systeme" (FKZ 081 5607)

Kernforschungszentrum Karlsruhe GmbH
ISSN 0303-4003

Kurzfassung

Ziel des DISCO-Projektes (Dezentralisierte Informationshaltungsstrukturen für Mini-Computersysteme) ist die Entwicklung einer auf Kleinrechnernetze zugeschnittenen verteilten Datenbankarchitektur und ihrer funktionellen Komponenten.

Die DISCO-Architektur basiert, im Gegensatz zu anderen Vorschlägen, auf einer logischen Dateiebene, die aus der Benutzersicht Transparenz bezüglich der Verteilung von Daten und Funktionen aufzeigt.

Es wurde das operationale Modell eines verteilten Dateisystems entwickelt, das als Schnittstelle eine logische Dateiebene anbietet. Es beinhaltet eine auf Interprozeßkommunikation beruhende Interaktion der funktionellen Komponenten und sieht Mechanismen zur Sicherung der operationalen Integrität verteilter Daten vor.

DISCO: An architecture for distributed databases

Abstract

The DISCO-project (Distributed Database for Small Computers) is aimed towards specification and implementation of a distributed data base architecture and its respective functional components for a minicomputer network environment.

In contrast to other proposals, the DISCO architecture is based on a logical file level providing transparency from a user's point of view with respect to distribution of functional components and data.

An operational model of a distributed file system offering a logical file level as an interface was developed, which relies on interprocess communication facilities and incorporates all access control functions necessary to provide operational integrity of stored distributed data.

Inhalt Seite

1. Einführung

1.1 Zielsetzung

Die Schaffung von Möglichkeiten zur Dezentralisierung der Datenhaltung in verteilten DV-Systemen entspricht dem Bedarf einer ständig wachsenden Zahl von Anwendern im Bereich der kommerziellen Datenverarbeitung und im technischen Bereich. Vor allem die Entwicklung leistungsfähiger, kostengünstiger Kleinrechner hat einen zunehmenden Trend zur Dezentralisierung von Datenverarbeitungskapazität in den genannten Gebieten verursacht; entsprechend müssen schritthaltend die Voraussetzungen für die Einrichtung adäquat konzipierter rechnergestützter Informationssysteme geschaffen werden.

Die Grundlage derartiger Systeme bilden Datenbanksysteme, deren Architektur an die Gegebenheiten der zum Einsatz kommenden verteilten DV-Systeme angepaßt ist. Eine Datenbankarchitektur für verteilte DV-Systeme kann die schrittweise Ausbaubarkeit eines auf dezentralisierte betriebliche Organisationsstrukturen zugeschnittenen rechnergestützten Informationssystems gewährleisten; die dezentralisierte Datenhaltung kommt erhöhten Anforderungen an Effizienz und Zuverlässigkeit durch ggf. redundante Informationsablage am Ort mit den häufigsten Zugriffsanforderungen entgegen.

Ein wichtiger Anwendungsbereich für dezentralisierte Datenbankarchitekturen sind arbeitsplatzorientierte, teilautonome Terminalsysteme, die über erhebliche Intelligenz und Speicherkapazität in den einzelnen Außenstationen verfügen. Derartige Systeme stellen heute nach übereinstimmender Ansicht der meisten Fachleute eine der am stärksten expandierenden Entwicklungen in der Datenverarbeitung dar.

Schwerpunkte für den Einsatz arbeitsplatzorientierter Systeme im industriellen Bereich sind Betriebsdatenerfassungs- und Steuerungssysteme. Man geht z.Z. davon aus, daß zukünftig etwa 70% - 80% aller DV-Investitionen im Prozeßlenkungs- und betrieblichen Dispositionsbereich für interaktive Terminalsysteme eingesetzt werden. Hervorzuheben ist dabei in dieser Anwendung die enge Verknüpfung zwischen den technisch-opera-

tiven, dispositiven und kaufmännischen Aspekten.

Auch im reinen Handels-, Banken- und Verwaltungsbereich setzt sich die arbeitsplatzorientierte Datenverarbeitung in Verbundsystemen durch. Netze mit in der Größenordnung von 10^4 Außenstationen sind, allerdings noch bei weitgehend zentraler Informationshaltung, im Aufbau.

Die Zielsetzung des als Initialvorhaben und "feasibility study" für weiterführende Arbeiten gedachten Vorhabens "Datenbankarchitektur für verteilte DV-Systeme" war es, entsprechend den oben angestellten Überlegungen, ein für den Aufbau und die Verwaltung verteilter Datenbanken in Rechnernetzen brauchbares Grundkonzept zu entwickeln und anhand experimenteller Studien zu untersuchen.

Als wesentliche Randbedingungen sollten in die Untersuchungen eingehen:

- Die als Arbeitsrechner in den Rechnernetzen in Betracht kommenden DV-Systeme sind Kleinrechner.
- Das zur Kopplung der Kleinrechner erforderliche Kommunikationssystem soll mit adäquatem Aufwand realisierbar sein und gleichzeitig die Arbeitsrechner von kommunikationsbedingten Verwaltungsaufgaben freihalten.
- Für die Interkommunikation zwischen Kleinrechnern sollten zunächst keine eigenen, grundsätzlich neuen Konzepte erarbeitet werden. Vielmehr sollte versucht werden, weitgehend auf im Großrechnerbereich entwickelte Konzepte zurückzugreifen, die sich zum Teil bereits im Vorfeld der Standardisierung befinden.
- Das zu entwickelnde Konzept sollte heterogene, verteilte Systeme berücksichtigen. Diese Randbedingung resultiert aus der Forderung nach Erweiterbarkeit und Einbeziehbarkeit bereits installierter DV-Systeme.
- Vorhandene konventionelle Betriebssysteme (Dateiverwaltungssysteme) sollten nach Möglichkeit als Basis für die Realisierung der zu entwickelnden Datenbankarchitektur eingesetzt werden können.

1.2 Vorgehensweise

Ein detailliertes, aber dennoch auf verschiedenen Zielrechnern effizient realisierbares (übertragbares) Konzept für die Architektur verteilter Datenbanken, das die unter 1.1 aufgeführten Randbedingungen berücksichtigt, setzt als Grundlage Schnittstellen für Interkommunikation und lokale Dateiverwaltung voraus, die gleichsam als systeminterner Standard auf unterschiedlichen, implementierungsbedingten Voraussetzungen realisiert werden können.

Schnittstellen, die die Formulierung der erforderlichen Kommunikationsabläufe zwischen funktionellen Komponenten eines verteilten Datenbanksystems ohne Kenntnis der implementierungsspezifischen Details des verwendeten Kommunikationssystems gestatten, standen zumindest für Großrechnernetze in Form von Schnittstellen für Interprozeßkommunikation (IPC) zur Verfügung.

Zunächst mußte gezeigt werden, daß eine äquivalent komfortable Interprozeßkommunikation auf den für den Großrechnerbereich entwickelten systematischen Ansätzen auch in verteilten Systemen mit Kleinrechnern verwirklicht werden kann, ohne daß dies mit einer erheblichen Leistungseinbuße erkauft werden muß. Damit war ein flexibles Nachrichtentransportsystem gefunden, das den Kommunikationsanforderungen verteilter Datenbanken in der vorgesehenen Umgebung genügt.

Die Einführung eines systeminternen Standards für die Schnittstelle zur lokalen Dateiverwaltung in den als Arbeitsrechner zum Einsatz kommenden Kleinrechnern mußte so erfolgen, daß einerseits die Abbildung der Standardschnittstelle auf eine Vielzahl heute verfügbarer Kleinrechnerbetriebssysteme ohne großen Aufwand möglich ist, andererseits die auf der Standardschnittstelle aufbauende Architektur verteilter Datenbanken in ihrem Leistungsvermögen nicht eingeschränkt wird.

Die Vorgehensweise bei der Konzipierung der zu erarbeitenden Architektur wurde durch die Entscheidung für die Verwendung interner Standardschnittstellen für Interkommunikation und lokale Dateiverwaltung entscheidend beeinflußt:
Als Grundstruktur eines operationalen Modells verteilter Daten-

banken war ein System von zu verteilten DV-Prozessen gruppierten funktionellen Komponenten, die nur über die Standardschnittstelle für Interprozeßkommunikation in Wechselwirkung treten können, vorgegeben. Der Grad der Redundanz der Realisierung dieser funktionellen Komponenten und ihre Verteilung sind variabel. Die Semantik der funktionellen Komponenten berücksichtigt die Anforderungen, die sich aus den der Standardschnittstelle für lokale Dateiverwaltung zu überlagernden höheren Funktionsebenen ergeben.

Bei der Konzipierung der höheren Funktionsebenen bot sich zunächst der "top-down-approach" an, ausgehend von konventionellen Architekturmodellen für zentralisierte Datenbanksysteme.
Die Untersuchung der Vielzahl von Möglichkeiten für die Einbringung des datenbezogenen Verteilungsaspektes in ein schichtenorientiertes Konzept verteilter Datenbanken, das durch die Benutzerschnittstelle auf der einen und die Standardschnittstelle für lokale Dateiverwaltung auf der anderen Seite abgegrenzt wird, zeigte sehr schnell, daß eine Datenhaltung in verteilten Systemen, die auf einer verteilungstransparenten logischen Dateiebene basiert, ein Höchstmaß an Flexibilität gewährleistet.

Auf einer logischen Dateiebene, die die Verwaltung verteilter Datenbestände vor den Benutzern bei Bedarf abschirmt, lassen sich Datenbasen unterschiedlicher Komplexität, die an die Anforderungen verschiedenster Anwendungen angepaßt werden können, unter Ausnutzung aller Vorteile der verteilten Datenhaltung aufbauen.

Ein Konzept für die Realisierung dieser logischen Dateiebene wurde, ausgehend von der internen Standardschnittstelle für die lokale Dateiverwaltung, "bottom-up" entwickelt. So wurden die durch die Kleinrechnerumgebung bedingten Restriktionen von Anfang an berücksichtigt und eine Beschränkung auf das Machbare erreicht.

Aus den oben aufgeführten Überlegungen resultiert unmittelbar die bei der Bearbeitung des Vorhabens angewendete Zuordnung

der Arbeiten zu drei Teilaufgaben:

- Entwicklung und Erprobung eines Nachrichtentransportsystems für die Interkommunikation zwischen den verteilten Komponenten eines Datenbanksystems.
- Untersuchung der Erweiterbarkeit konventioneller Ansätze und Realisierungen der Datenverwaltung sowie Ableitung ggf. alternativer Architekturkonzepte für verteilte Datenbanken unter besonderer Berücksichtigung von Kleinrechnersystemen.
- Konzipierung und Einzelerprobung eines operationalen Architekturmodells für verteilte Datenbanken mit Kleinrechnern.

Den beiden ersten Teilaufgaben ist gemeinsam, daß sie auf umfangreichen Analysen des Standes der Technik basieren und mit ihren Ergebnissen Grundlage des zu erarbeitenden Konzeptes für eine Datenbankarchitektur für verteilte DV-Systeme bilden.

Ziel der ersten Teilaufgabe war es, ein spezifisches Nachrichtentransportsystem mit einer Schnittstelle für Interprozeßkommunikation zur Erprobung wesentlicher Funktionen der dritten Teilaufgabe zu entwickeln und auf einem zu installierenden Kleinrechnerverbund schrittweise zu implementieren.

Gleichzeitig war die Benutzung bereits in der Standardisierung befindlicher elementarer Kommunikationsprotokolle (HDLC) vorgesehen.

Innerhalb der zweiten Teilaufgabe sollten im Einsatz befindliche bzw. vorgeschlagene Datenmodelle, Dateisysteme und Datenbanksysteme auf ihre Eignung für die Einbeziehung in ein Architekturkonzept verteilter Datenbanken untersucht werden.

Dabei sollten, unabhängig von der Einschränkung auf kleine Systeme, auch allgemeine Fragen der Verteilung und Mehrfachhaltung von Daten und Funktionen untersucht werden.

Ziele der in dieser Teilaufgabe beinhalteten Konzeptstudien waren Analysenergebnisse, die auf in der dritten Teilaufgabe zu untersuchende Fragestellungen, Grundkonzepte und ggf. Entwurfsalternativen für eine für verteilte DV-Systeme geeignete Datenbankarchitektur hinleiten.

Als Vorstufe zur Implementierung und Einzelerprobung sollte

in der dritten Teilaufgabe ein operationales Modell einer verteilten Datenbankarchitektur erarbeitet werden, das die zur Realisierung erforderlichen funktionellen Komponenten identifiziert, gruppiert und ihre über das Nachrichtentransportsystem realisierten Wechselwirkungen wiedergibt.

Spezielle Aufmerksamkeit sollte dem Problem der Zugriffskoordinierung zuteil werden. Ziel waren Verfahren, die die Haltung redundanter Datenbestände bei gleichzeitiger Gewährleistung der Konsistenz gestatten und dadurch den Aufbau hochverfügbarer, fehlertoleranter Systeme ermöglichen.

Der dem Vorhaben gegebene Charakter einer "feasibility study" mit engen Schranken für Zeit und Aufwand gebot es, den neben der Frage nach der Machbarkeit verteilter Datenbanken sekundären Aspekt des Datenschutzes in derartigen Systemen zunächst unberücksichtigt zu lassen. Die Behandlung der technischen Seite dieser, aufgrund jüngster Aktivitäten des Gesetzgebers besonders aktuellen Frage, sollte in einem gesonderten Vorhaben behandelt werden.

1.3 Darstellung der Ergebnisse

Im vorliegenden Bericht werden die bei der Bearbeitung des Vorhabens erzielten Ergebnisse nicht streng den einzelnen Teilaufgaben zugeordnet. Vielmehr orientieren sich Form und Reihenfolge der Darstellungen an der Rolle der Ergebnisse bezüglich des Gesamtkomplexes "verteilte Datenbanken".

Zunächst wird in Kapitel 2 der Problemkreis der Datenhaltung in verteilten DV-Systemen vor dem Hintergrund des Standes der Technik einführend betrachtet.

In Kapitel 3 folgt dann die Vorstellung des im Rahmen des Vorhabens entwickelten Grobkonzepts für verteilte Datenbankarchitekturen, das sich stark an für konventionelle Datenbanksysteme entwickelte Konzepte anlehnt /ANS1, DIT1, SAA1, SCH5/. Im Vordergrund steht die Diskussion möglicher Varianten der Verteilung von Daten und die Bewertung dieser Varianten. Die Diskussion führt zur verteilungstransparenten logischen Dateiebene als ideale Basis für verteilte Datenbanken auf

Kleinrechnersystemen.

Kapitel 4 dient der Einführung eines für die Realisierung logischer Dateiebenen in verteilten DV-Systemen auf der Grundlage einer internen Standard-Schnittstelle für lokale Dateiverwaltung entwickelten Mehrschichtenkonzeptes und dem Nachweis der Brauchbarkeit dieses Konzeptes als Grundlage für alle Datenbanksysteme, die auf den gängigen Datenmodellen basieren.

Der Umsetzung des in Kapitel 4 diskutierten Konzeptes für die Verteilungen der Daten in ein operationales Modell mit verteilten funktionellen Komponenten unter Einbeziehung der über das Nachrichtentransportsystem verwirklichten Interprozeßkommunikation dient Kapitel 5.

Datenhaltung in verteilten Systemen stellt aufgrund der möglichen parallelen Transaktionsbearbeitung bei verteilten Funktionen und verteilten Daten besondere Anforderungen an Verfahren zur Sicherung der Integrität der abgespeicherten und bei Abfragen gewonnenen Informationen. Hierzu wurden die in Kapitel 6 beschriebenen Techniken entwickelt, die auf speziellen Formen der Interkommunikation über das Nachrichtentransportsystem beruhen. Besondere Berücksichtigung bei der Entwicklung der diskutierten Verfahren fand die redundante Datenhaltung und die damit mögliche Steigerung der Systemverfügbarkeit.

Kapitel 7 schließlich geht auf das für die Realisierung einer systeminternen Standard-Schnittstelle für Interprozeßkommunikation gewählte, auf Ergebnissen von Untersuchungen der im Großrechnerbereich zu findenden Ansätze beruhende Konzept ein. Als greifbares Ergebnis wird die Pilotimplementierung eines entsprechenden Nachrichtentransportsystems auf einem am Institut für Datenverarbeitung in der Technik realisierten verteilten DV-System, einem aus Mini- und Mikrorechnern aufgebauten lokalen Rechnernetz beschrieben.

2. Datenhaltung in verteilten DV-Systemen

2.1 Verteilungsbegriff

Die Bezeichnung "Verteiltes DV-System" entspricht z.Z. noch keinem normierten Begriff; als Sprachregelung auf internationaler Ebene hat sich jedoch eine, wenn auch nicht sehr präzise Begriffsbestimmung, herausgebildet: Unter "distributed systems" werden DV-Systeme verstanden, die unter der Kontrolle eines übergeordneten dezentralisierten Betriebssystems den koordinierten paralleln Ablauf von DV-Prozessen auf physikalisch/räumlich verteilten, lose gekoppelten (asynchron arbeitenden) Prozessoren gestatten.
Diese Begriffsbestimmung eliminiert somit klassische, ausschließlich über einen gemeinsamen Hauptspeicher kommunizierende Mehrprozessorsysteme ebenso, wie Rechnernetze mit zentralisierter Betriebsmittelverwaltung.

Für die Datenhaltung in verteilten DV-Systemen wird neben einer Zentralisierung aller Datenverwaltungsfunktionen auf einem ausgezeichneten, spezialisierten Rechner - das DATACOMPUTER-Projekt ist hierfür ein herausragendes Beispiel /MAS1/ -, heute eine zweite Alternative verfolgt, die nicht nur den Zugriff zu Daten, sondern auch deren freie Verteilung innerhalb eines verteilten DV-Systems anstrebt ("distributed data base").

Anpassung der EDV an Organisationsstrukturen und nicht, wie heute üblich, Anpassung der Organisationsstrukturen an die EDV, erhöhte Zuverlässigkeit und Verfügbarkeit und Wirtschaftlichkeitsüberlegungen, die auf dem Trend zu wesentlich stärker fallenden Rechnerkosten im Vergleich zu Übertragungskosten beruhen, sind die gewichtigsten Argumente für die Einrichtung verteilter Datenbestände.

Unter Verteilung soll die physikalische Zuordnung von Verteilungsobjekten, d.h. von Daten, Funktionen, Transaktionen und Benutzern, zu den Rechnern eines verteilten DV-Systems verstanden werden. Für diese Zuordnung bieten sich mehrere Möglichkeiten an, abhängig davon, ob

- eine Zerlegung von Verteilungsobjekten in Teilobjekte, die

selbst wieder Verteilungsobjekte sein können, durchgeführt werden kann,

- Verteilungsobjekte nur einem Rechner oder, redundant realisiert, mehreren Rechnern simultan zugeordnet werden sollen.

Redundante Realisierung (Kopie) und Zerlegung sind die Instrumente, mit denen Verteilung gesteuert werden kann.

Beim Verteilungsobjekt Daten (hierunter fallen neben Primärdaten auch Zugriffspfade, Kataloge und Schemata) wird die Art der Verteilung von der Form der Datenhaltung wesentlich beeinflußt: Für satzweise zu Dateien organisierte Daten kann zwischen einer grob granulierten Verteilung auf Dateiebene und einer feinen Granulierung auf Satzebene gewählt werden. In Datenbanksystemen treten tiefer strukturierte Datenobjekte auf, so daß mehrere Granulierungsstufen unterschieden werden können. So ist z.B. in relationalen Datenbanksystemen eine Verteilung von Relationen, Tupeln, Attributen oder gar von einzelnen Attributwerten denkbar. Bei einer redundanten Realisierung von Daten muß die Identität (Konsistenz) der einzelnen Kopien gewährleistet werden.

Bei Funktionen als Verteilungsobjekten wird eine Zerlegung in Teilfunktionen dadurch begrenzt, daß diese letztlich auf Prozesse abgebildet werden müssen. Eine redundante Realisierung von Funktionen erfordert deren Koordinierung, wenn sie auf gemeinsame Resourcen zugreifen. So kann z.B. in einem verteilten Datenverwaltungssystem die Teilfunktion, die Sperrmechanismen realisiert, abgespalten und einem (zentrale Kontrolle) oder mehreren Rechnern (dezentrale Kontrolle) zugeordnet werden.

Für Transaktionen als Verteilungsobjekte ist deren Zerlegung in parallel ausführbare Teiltransaktionen aus Gründen der Effizienzsteigerung wünschenswert, während redundante Transaktionsausführung bis auf wenige Spezialfälle (z.B. in "stand-by"-Systemen) eine untergeordnete Rolle spielt.

Bezüglich der Gesamtheit der Benutzer ist in der Regel eine aufgabenorientierte Aufteilung in Benutzergruppen und eine

organisatorische Zuordnung zu den Rechnern eines verteilten Systems sinnvoll. Funktionsbedingte Mehrfachzuordnungen sind erforderlich, wenn z.B. ein Benutzer gleichzeitig an mehreren Aufgaben partipiziert.

Verteilung von Daten, Funktionen, Transaktionen und Benutzern kann nicht isoliert voneinander betrachtet werden. Vielmehr existieren wechselseitige Abhängigkeiten, die die Vielfalt der Verteilungsmöglichkeiten einschränken. Liegt z.B. die Verteilung von Daten fest, so kann über die Verteilung der erforderlichen Zugriffsfunktionen nicht mehr frei verfügt werden.

2.2 Verteilungskriterien

Kriterien, die die Zuordnung der Verteilungsobjekte Daten, Funktionen, Transaktionen und Benutzer zu den Rechnern eines verteilten DV-Systems sowie den Grad der Zerlegung und der redundanten Realisierung beeinflussen, können im Idealfall als Zielgrößen bzw. Parameter von Optimierungsmodellen spezifiziert werden.

Formale Modelle, die eine an Verteilungskriterien orientierte optimale Verteilung zu ermitteln gestatten, existieren jedoch bisher nur für Spezialfälle (vgl. 2.7). Insbesondere das gleichzeitige Optimieren nach mehreren Kriterien, das Formalisieren von Begriffen, wie Organisationsstruktur und der Einfluß von Datenschutzbetrachtungen auf die Verteilung stellen ungelöste Probleme dar. Es sollen daher nur qualitative Betrachtungen angestellt werden, welche Verteilungskriterien berücksichtigt und ggf. mit den Instrumenten der Zerlegung und redundanten Realisierung optimiert werden können:

- Anpassung an die Organisationsstruktur der Systemumgebung:
  Besonders bei dezentraler Organisationsstruktur ist eine Anpassung des zugehörigen DV-Systems in der Art anzustreben, daß das Gesamtsystem in entsprechende autonome Teilsysteme aufgegliedert wird. Dies kann durch eine Zerlegung in funktionsverschiedene Komponenten (z.B. entsprechend der organisatorischen Gliederung eines Unternehmens in Produktionsabteilung und kaufmännische Abteilung) oder in funk-

tionsidentische Komponenten (z.B. für Banken mit mehreren Filialen) erreicht werden.

- Datenschutz:
  Der Einfluß von Datenschutzanforderungen auf die Verteilung ist noch nicht hinreichend geklärt. Bisweilen wird das Aufteilen schutzwürdiger Daten auf verschiedene Rechner eines verteilten DV-Systems schon als Maßnahme zur Erhöhung des Datenschutzes vorgeschlagen /SCH1/.

- Erhöhung der Zuverlässigkeit und Verfügbarkeit:
  Die in einem verteilten DV-System durch das Vorhandensein mehrerer autonomer Teilsysteme erhöhte Zuverlässigkeit und Verfügbarkeit kann durch das Instrument der redundanten Realisierung gezielt vergrößert werden.

- Minimierung der Kosten:
  Die Kosten der Datenhaltung werden primär durch die von der Zahl der Zugriffe abhängigen Übertragungskosten, die Speicherkosten und die Programmlaufkosten bestimmt. Hier existieren bereits Optimierungsmodelle, die die Bestimmung der kostengünstigsten Verteilung von Daten, Funktionen und Transaktionsbearbeitungen gestatten; das Optimum kann mit den Instrumenten der Zerlegung und der redundanten Realisierung eingestellt werden.

- Optimale Kapazitätsnutzung:
  Zur bestmöglichen Nutzung der vorhandenen Kapazitäten von Übertragungsleitungen, Speichern, Prozessoren und Softwarekomponenten kann ein Spektrum von Verteilungsmöglichkeiten für Daten, Funktionen, Transaktionen und Benutzern eingesetzt werden. Die Grenzkapazitäten müssen bei der Auslegung der Hardware durch eine Grobanalyse des zu erwartenden Systemverhaltens bestimmt und bei der Verteilungsoptimierung als Restriktionen berücksichtigt werden.

- Minimierung der Antwortzeit:
  Eine Minimierung der Antwortzeit ist durch Zerlegung der Datenobjekte in anfragerelevante Teilobjekte und die redundante Realisierung dieser Teilobjekte einschließlich der zugehörigen Funktionen erreichbar, falls dies eine Zer-

legung der Transaktionen in parallel ausführbare Teiltransaktionen induziert /GHO1/ und die Kommunikationsverzögerungen geringer als die Bearbeitungsdauer der Teiltransaktionen sind.

2.3 Lokalisierung von Datenbeständen

Lokalisierung von Datenbeständen bei verteilter Datenhaltung umfaßt das Identifizieren und Auffinden von benannten Objekten im verteilten DV-System.

Sind die benannten Objekte Dateien, identifiziert durch genau einen Namen, so kann ein als Katalog bezeichnetes Lokalisierungsschema eingeführt werden, das für jedes Objekt einen Eintrag enthält mit Verweis auf den Rechner und Datenträger, auf dem das Objekt abgelegt ist, zusammen mit einem weiteren Katalog, dem Inhaltsverzeichnis des Datenträgers. Dieses liefert dann die exakte Adresse.

In verteilten DV-Systemen können Katalogorganisationen entweder in herkömmlicher zentralisierter Form oder dezentral angelegt werden. Zentralisierte Lösungen werden meist Verfügbarkeits- und Effizienzanforderungen entgegenstehen. Dezentralisierte Lösungen bieten dagegen die Möglichkeit der Parallelisierung von Suchvorgängen bei erhöhter Verfügbarkeit des gesamten Katalogsystems.

Varianten dezentralisierter Katalogorganisationen mit unterschiedlichem Grad an redundanter Realisierung sind denkbar:

- Ist die Zahl der zu erfassenden benannten Datenobjekte gering, so ist die mehrfache Realisierung eines globalen Katalogs, der die Einträge für sämtliche Dateien des verteilten DV-Systems enthält, vertretbar.
- Bei einer großen Anzahl zu erfassender Datenobjekte bietet sich dagegen die Einrichtung einer hierarchischen Katalogstruktur mit auf die lokalen Kataloge der einzelnen Rechner verweisenden Teilkatalogen der obersten Hierarchiestufe an. Diese Teilkataloge können redundant realisiert und auf mehreren Rechnern des verteilten DV-Systems geführt werden.

Bei einem solchen hierarchischen Katalogsystem wird eine festgelegte Teilzeichenreihe der Bezeichnung des Datenobjekts als Schlüssel für den redundanten Teilkatalog an der Spitze der Kataloghierarchie verwendet. Häufig wird auch für die Objektbezeichnungen eine abgestufte Namensgebung ("qualified names") vorgeschlagen /HEF1, HOD2/. Eine andere Variante für eine solche Katalogstruktur besteht darin, den Teilkatalog der obersten Hierarchiestufe durch eine Berechnungsvorschrift zu ersetzen.

Bei der Realisierung dezentralisierter Katalogorganisationen mit redundanten Teilkatalogen müssen besondere Maßnahmen zur Konsistenzerhaltung getroffen werden; beim Wiederanlauf ausgefallener Rechner ist für die Wiedereinrichtung der betroffenen lokalen Kataloge zu sorgen.

2.4 Integrität und parallele Zugriffe

Die Integrität eines Datenbestandes umfaßt im weitesten Sinne /BAY1/ die Forderung nach Vollständigkeit, Fehlerfreiheit, Folgerichtigkeit und Vertraulichkeit der Daten. Sie ist insbesondere bei inhaltsverändernden Zugriffen gefährdet.

In der Regel bestehen zwischen den Daten eines Datenbestandes innere Beziehungen, die z.B. resultieren aus

- Vorschriften seitens der Umwelt über strukturelle Abhängigkeiten
- der Forderung nach semantischer Identität redundant realisierter Daten.

Diese inneren Beziehungen sind in Konsistenzregeln /EWS2, SCH4/ zusammengefaßt. Der Zustand eines Datenbestandes heißt konsistent, falls die Konsistenzregeln simultan erfüllt sind.

Um Maßnahmen des Systems zur Integritätssicherung zu unterstützen, wurde die Transaktion als Einheit eingeführt. Nur innerhalb einer Transaktion darf auf Datenbestände zugegriffen werden. Die Menge der Daten, die in ihrer Gesamtheit in einem für eine Transaktion konsistenten Zustand sein muß, bezeichnet man als Konsistenzbereich.

Bei inhaltsverändernden Zugriffen können folgende Integritätsverletzungen auftreten /BAY1/:

- Verletzungen der semantischen Integrität: Transaktionen ändern Daten fehlerhaft oder berücksichtigen die Konsistenzregeln nicht.
- Verletzungen der operationalen Integrität: Unterschiedliche Transaktionen, die bei serieller Abarbeitung die semantische Integrität nicht verletzen würden, können sich bei parallelem, unkontrolliertem Zugriff auf Datenbestände stören, falls ihre Konsistenzbereiche sich überlappen (Mehrbenutzerumgebung).

Insbesondere die Sicherung der operationalen Integrität wirft im Zusammenhang mit der Datenhaltung in verteilten Systemen Probleme auf: Die parallele Abwicklung von Transaktionen bei verteilten, ggf. redundant realisierten Datenbeständen, wird zur normalen Betriebsform.

Zur Gewährleistung der operationalen Integrität müssen Sperrmechanismen zur Zugriffskontrolle eingerichtet werden. Die Entwicklung geeigneter Sperrmechanismen ist sowohl für zentral gehaltene als auch verteilt geführte Datenbestände fundamental. Im letzten Fall können jedoch die je nach Anwendung im Vordergrund stehenden Verteilungskriterien eine dezentrale Organisation der Sperrmechanismen implizieren.

2.5 Interkommunikation

Die Verteilung von Datenverwaltungsfunktionen und Datenbeständen sowie die Verwirklichung verteilter Zugriffsmöglichkeiten und einer damit gekoppelten verteilten Transaktionsbearbeitung erfordert eine geeignete Einrichtung für den Austausch von Informationen zwischen den resultierenden funktionellen Komponenten.

Ein Nachrichtentransportsystem, das dieser Aufgabe gerecht wird, muß drei Arten des Informationsaustauschs bewältigen:

- Austausch von Kontrollnachrichten
- Übertragung von Dateien ("file transfer")

- gezielter Transfer vorgegebener Datensätze bei abgesetztem Zugriff ("remote access")

Der Austausch von Kontrollnachrichten dient der Aktivierung und Synchronisation. Die Übertragung von Dateien zwischen den Arbeitsrechnern eines verteilten Systems und die Einrichtung von Datenbestandskopien wird im Rahmen der Datensicherung und aus Effizienzüberlegungen heraus erforderlich. Die häufigste Form des Datentransfers in transaktionsorientierten Datenverwaltungssystemen ist die Übertragung einzelner Datensätze bei abgesetztem Zugriff vom Ort des Datenbestandes zum Ort der Transaktionsbearbeitung und umgekehrt.

Die Realisierung der für die Datenhaltung in verteilten Systemen benötigten funktionellen Komponenten geschieht durch deren Zuordnung zu einem System verteilter Prozesse; entsprechend bietet ein Nachrichtentransportsystem mit Schnittstelle für Interprozeßkommunikation (IPC) eine geeignete Grundlage, auf der der geforderte Informationsaustausch realisiert werden kann /ABS1, WAL1/. Schnittstelle für Interprozeßkommunikation bedeutet, daß für beliebig verteilte Prozesse, unabhängig von der Art der Realisierung des physikalischen Datentransports, der Austausch von Nachrichten spezifiziert werden kann. Das Senden und Empfangen von Nachrichten erfolgt dabei über eindeutig identifizierte Ein- und Ausgänge der Prozesse, sog. "Ports". Die zur Identifikation verwendeten Portnamen müssen - wesentlichste Restriktion - auf den vom Nachrichtentransportsystem verwalteten Namensraum abbildbar sein.

IPC-Funktionen für Senden, Empfangen, Synchronisation, für die Verwaltung des Namensraums und die Zuordnung von Ports ermöglichen die Formulierung anwendungsspezifischer Dialoge durch Spezifikation von "Protokollen", Konventionen für die Abwicklung des Nachrichtenaustauschs. Die übertragbare Realisierung der drei obengenannten Arten des Informationsaustauschs reduziert sich somit auf Zusammenstellung spezifischer Kommunikationsprotokolle auf IPC-Ebene.

2.6 Heterogenität

Die Einbeziehung nicht kompatibler Rechnersysteme in verteilte DV-Systeme bewirkt für Interkommunikation und Datenhaltung eine Vielzahl von Anpassungsproblemen.

Integrierte Datenhaltung in verteilten Systemen, gekennzeichnet durch eine einheitliche Endbenutzerschnittstelle, erfordert eine verteilungstransparente Datenverwaltungsschnittstelle, die im Falle heterogener Systeme auf unterschiedliche lokale Dateiverwaltungen abzubilden ist, wenn auf bereits vorhandenen Betriebssystemen aufgesetzt werden soll.

Nur in den seltensten Fällen sind Schnittstellen und Organisationsstrukturen kompatibel. Durch unterschiedliche Sprachimplementierungen wird darüber hinaus der Aufwand für die Einbettung der Datenverwaltungsschnittstelle in Programmiersprachen vervielfacht.

Inkompatibilitäten der von den individuellen Betriebssystemen unterstützten Dateiverwaltungen sind in der Regel bereits durch unterschiedliche Rechnerstrukturen, die in verschiedenen Datendarstellungen resultieren sowie durch verschiedenartige physikalische Strukturierung der Speichermedien gegeben.

Für den Bereich der Interkommunikation hat man es durch frühzeitige Standardisierung von Hardwareschnittstellen und Leitungsprotokollen verstanden, zumindest für den hardwarenahen Bereich, die Zahl der erforderlichen Anpassungen zu beschränken. Für eine Schnittstelle zur Interprozeßkommunikation existieren jedoch noch keinerlei anerkannte Standards.

Wie im Bereich der Interkommunikation stellt auch für die Datenverwaltung die Definition von Standardschnittstellen den effizientesten Weg zur Kompatibilisierung dar; der Weg kann jedoch hier nur über die Anlehnung der festzulegenden systeminternen Schnittstellen an die am häufigsten auftretenden gemeinsamen Merkmale heute verfügbarer Datenverwaltungssysteme erfolgen.

2.7 Einbeziehung von Kleinrechnern

Hauptgrund für den derzeitigen Trend zu verteilten DV-Systemen ist das umfangreiche Angebot an leistungsfähigen, kostengünstigen Kleinrechnern. Die durch eine dezentrale Hardware und den Wegfall von aufwendigen Rechenzentren erreichbare Benutzernähe, der durch dedizierte Soft- und Hardware reduzierte Systemoverhead sowie die leichte Erweiterbarkeit und damit Anpaßbarkeit an wachsende Aufgaben sind die gewichtigsten Argumente für das Ersetzen komplexer Zentralsysteme durch Kleinrechnernetze, insbesondere wenn dies bei geringerem Investitionsaufwand möglich ist /BER1/.

Bei der Konzipierung verteilter Datenverwaltungssysteme auf der Grundlage von Kleinrechnersystemen muß man die durch die Eigenheiten der Kleinrechnerarchitektur bedingten Beschränkungen bzgl. Hauptspeicherkapazität und Leistungsfähigkeit berücksichtigen, auch wenn für die Zukunft ein Trend zu größeren und leistungsfähigeren Systemen erkennbar ist. Diesem Umstand ist durch Reduktion des Verwaltungsaufwandes, z.B. durch Minimierung des Konversions- und Kompressionsaufwandes, Begrenzung von Pufferspeichern und Beschränkung der Mächtigkeit von Datenverwaltungsoperatoren Rechnung zu tragen.

Zur Reduktion des Verwaltungsaufwandes trägt auch die Auslagerung von häufig angesprochenen Funktionen auf spezielle Prozessoren bei, eine Möglichkeit, die durch den Einsatz von Mikroprozessoren leicht zu realisieren ist. Dies gilt insbesondere für die Auslagerung von Kommunikations- und Koordinationsfunktionen (vgl. Kap. 6 und 7). In /BER1/ wird für die nahe Zukunft ein Trend auch zur Auslagerung spezifischer Datenbankoperatoren auf spezielle Datenbankprozessoren prognostiziert.

Die für Kleinrechner angebotenen, relativ ausgereiften, Dateiverwaltungssysteme können als abgemagerte Versionen der für die meisten Großrechner angebotenen Systeme eingestuft werden; die Entwicklung von Datenbanksystemen für Kleinrechner ist dagegen erst in den Anfängen. Als konsequente Vorgehensweise

bei der Konzipierung einer Architektur verteilter Datenverwaltungssysteme empfiehlt es sich daher, die lokalen Dateiverwaltungssysteme als Ausgangsbasis zu verwenden ("bottom-up-approach"). Dadurch wird die Entwicklung und Pflege redundanter Systembausteine vermieden.

2.8 Stand der Technik

Die Technologie verteilter DV-Systeme hat in den letzten Jahren eine beachtliche Weiterentwicklung erfahren. Gradmesser der Entwicklung ist das Maß, in dem der Begriff "Verteiltes DV-System" die alten Begriffe Rechnernetz und Mehrrechnersystem ersetzt: Im Vordergrund des derzeitigen Interesses stehen nicht mehr die elementare Interkommunikation zwischen autonomen Rechnersystemen und der damit mögliche Aufbau vermaschter, vorwiegend jedoch hierarchischer, zentralisierter Rechnerverbunde, sondern Fragen des "resource sharing", der verteilten Betriebsorganisation und der Interprozeßkommunikation. Die Lösung dieser Probleme schafft die Möglichkeit, verteilte DV-Systeme im Sinne der in 2.1 gegebenen Definition aufzubauen.

Die Datenhaltung in verteilten DV-Systemen als Problem der Resource-Verwaltung nahm bisher in den meisten Konzepten gegenüber anderen Problemen dieser Art keine Sonderstellung ein: Datenverwaltungssysteme wie Datenbank- und Dateisysteme wurden als zentral verwaltete "resources" im verteilten DV-System realisiert; allenfalls wurde eine Verteilung der Datenbasis unter Beibehaltung zentralisierter Zugriffskontrollfunktionen berücksichtigt /CHA1, KAR1/.

Redundante Datenhaltung wurde bisher nur vereinzelt in Konzepte aufgenommen, Beispiel hierfür ist INPOL /KAR1/.
Eine inhaltsorientierte Aufteilung von Daten schlägt das Konzept für eine in der Planung befindliche, verteilte Version von INGRES vor /STN1, STO1/.

Kriterien, die für die Verteilung von Datenbeständen herangezogen wurden, waren

- die organisatorische Struktur der Systemumgebung
- Kosten- und Antwortzeitoptimierung.

Verfügbarkeit und Zuverlässigkeit spielten, bedingt durch die zentral oder hierarchisch orientierte Architektur, eine nur untergeordnete Rolle. Entsprechend blieben die mittels redundanter Datenbestände zu lösenden Probleme wie Fehlertoleranz und Wiederanlauf weitgehend unberücksichtigt.

Für die Kosten- und Antwortzeitoptimierung wurde eine Reihe von Optimierungsmodellen entwickelt, die zum Teil neben der Verteilung von Daten die variable Plazierbarkeit der zugehörigen Verwaltungs- und Verarbeitungsprogramme, d.h. die Verteilung von Funktionen, vorsahen /CAS1, CHU3, ESW1, GHO1, HOL2, MAR1, WHI1/.
Neuere Arbeiten auf dem Gebiet der Optimierungsmodelle ermöglichen, ausgehend von einer vorgegebenen Verteilung für Datenbestände und Funktionen, die Ermittlung der günstigsten rekonfigurierten Verteilung aus dem aktuellen Betriebszustand eines verteilten Systems heraus /LEM2/.

Verteilte Datenhaltung mit redundanter Realisierung von Datenbeständen erfordert Sperrmechanismen, die die operationale Integrität auch bei parallelen Zugriffen gewährleisten.

Für zentral gehaltene Datenbestände wurden leistungsfähige Sperrmechanismen entwickelt /BAY1/, die auch für verteilte Datenbestände mit zentraler Zugriffskontrolle eingesetzt werden können.

Die Mechanismen ermöglichen

- die unterschiedliche Granulierung der Sperrungen von Daten zur Maximierung der Anzahl parallel bearbeitbarer Transaktionen /GRA1/,
- die Sperrung von Datenmengen über deren Inhalt zur Verhinderung der Materialisation von Phantomdaten /ESW2/.

Konsistenzsichernde Verfahren für dezentralisierte Zugriffskontrolle durch ein System funktionell äquivalenter, gleichberechtigter Kontrollinstanzen existieren bisher lediglich für den Spezialfall redundant realisierter Dateien /HOD1/. Diese Verfahren berücksichtigen keine Kooperation von Kontrollinstanzen, falls Transaktionen auf strukturell abhängige Daten im Zuständigkeitsbereich unterschiedlicher Kontrollinstanzen

zugreifen.

Bei bestehenden Systemen wurden bisher völlig unterschiedliche, von Anwendungen diktierte Forderungen an den Grad der Konsistenz gestellt /CJM1, JOT1, KAR1, MIL1/. Die Konsequenz daraus war eine Vielfalt spezieller Synchronisationsmechanismen.

Weitere Untersuchungen haben der Realisierung von netzweiten Adressierungsschemata gegolten. Teilweise werden diese Schemata auf Speicherniveau (Blockebene) durch Definition eines globalen linearen Adreßraumes, teilweise auf Namensebene mit Hilfe von Katalogen realisiert.

Da eine Beschränkung auf Speicherebene viele Vorteile, die den Einsatz von verteilten DV-Systemen motivieren, z.B. redundante Realisierung von Datenbeständen, Austauschbarkeit von Rechnern und Datenträgern, nicht berücksichtigen, konzentrieren sich die Überlegungen auf die Realisierung von Katalogsystemen unterschiedlicher Redundanz und Organisation. Mehrere Möglichkeiten werden in /CHU1/ behandelt, hierarchische Katalogstrukturen werden in /MIL1, HEI1, HOD2, STN1/ beschrieben.

Die Interprozeßkommunikation steht als Instrument der Interkommunikation zwischen den funktionellen Komponenten eines verteilten Datenverwaltungssystems, zumindest in Großrechnernetzen, grundsätzlich zur Verfügung /AUE1, ABS1, CER1, CHM1/. Bei der Einbringung zentral verwalteter, autonomer Datenverwaltungssysteme als "sharable resource" in verteilte DV-Systeme konnte jedoch auf den Einsatz der Interprozeßkommunikation verzichtet werden, solange bei abgesetzter Benutzung (etwa über ein Terminal) lediglich eine lokale Benutzung simuliert wurde.

Mittlerweile wurden Systeme konzipiert und realisiert, die die Transaktionsbearbeitung am Ort der Transaktionsgenerierung vorsehen /CHA1, CHU3, CJM1, HEI1, HPT1, KAR1, LAL1, MIL1, PEE1, STO1/. Hier ist der Einsatz von "remote access"- und "file transfer"-Protokollen, die auf der Interprozeßkommunikation aufbauen, erforderlich.

"File transfer"- und "remote access"-Protokolle wurden bereits für eine Reihe von Installationen realisiert /CHM1, DEC1, HEB1, HPT1/, eine detaillierte Spezifikation von IPC, "file transfer" und "remote access" auf der Grundlage der paketorientierten Schnittstelle für Interrechnerkommunikation X.25 /CCI1/ wird zur Zeit von der X.25-Benutzer-Arbeitsgemeinschaft PIX erarbeitet /SAB1/; Zielsysteme sind jedoch auch hier Großrechnernetze.

Die Einrichtung einer integrierten Datenhaltung in heterogenen verteilten DV-Systemen wurde bisher nur in Einzelfällen versucht. Beispiele sind INPOL und das Pilotprojekt SOCRATES /CHS1, CHU2, KAR1/. Das Problem der Anpassung wird dabei durch eine weitgehend autonome Verwaltung von Teildatenbeständen entschärft.

Die Datenhaltung in verteilten Kleinrechnersystemen ist bisher in der ersten Konzipierungsphase steckengeblieben; dies gilt insbesondere für die Entwicklung von Datenbanksystemen, die eine Verteilung von Datenbeständen und Funktionen zulassen.
Die Gründe dafür liegen in der zur Zeit noch begrenzten Leistungsfähigkeit von Kleinrechnersystemen.

3. Architekturkonzepte für verteilte Datenbanken

3.1 Konzipierung auf der Basis des ANSI/X3/SPARC-Vorschlags

Der Ansatz für die Entwicklung von Architekturkonzepten für verteilte Datenbanken besteht in der Einbettung des Verteilungsaspektes in den ANSI/X3/SPARC-Vorschlag /ANS1/. Dieses Konzept ist aus Standardisierungsbestrebungen für Datenbankarchitekturen abgeleitet und wird mittlerweile allgemein akzeptiert. Es sieht - aus konzeptuellen Gründen - eine schichtenförmige Aufgliederung eines Datenbanksystems in die folgenden vier Ebenen vor (Bild 3.1):

- die Ebene der externen Modelle zur Modellierung der Datenbasis aus der Sicht einzelner Benutzer oder Benutzergruppen - Benutzersicht,
- die Ebene des konzeptionellen Modells als Vereinigung aller möglichen Benutzersichten - logische Systemsicht,
- die Ebene des internen Modells zur Modellierung der Einrichtungen, die die Ablage der Datenbasisobjekte auf den Speichermedien beschreiben und ein schnelles Auffinden dieser Objekte ermöglichen (Zugriffspfade) - interne Systemsicht,
- die Ebene des Datei-Modells zur Beschreibung der Datenbasis als eine Menge von Dateien, die in unterschiedlicher Organisationsform auf den Hintergrundspeichern abgelegt werden können - Datei-Sicht.

Die Beschreibung der zwischen den einzelnen Ebenen notwendigen Transformationen ist in den verschiedenen Abbildungen (vgl. Bild 3.1) angesiedelt. Die Durchführung dieser Transformationen wird unter Einbeziehung der entsprechenden Abbildungsinformationen in den zwischen den Ebenen angeordneten Schichten vorgenommen.

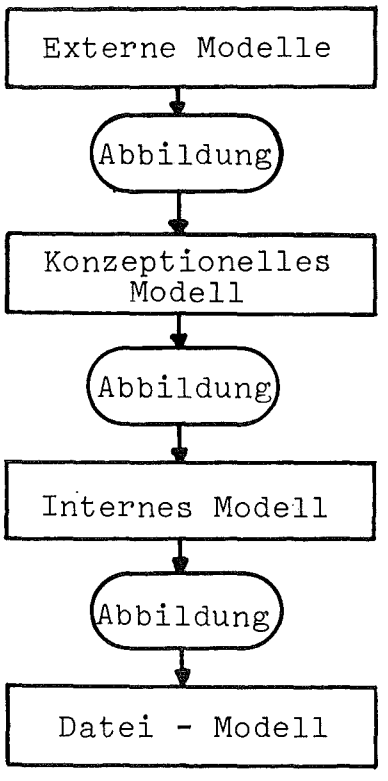


Bild 3.1: Ebenen des ANSI/X3/SPARC-Vorschlags

Die Einbettung des Verteilungsaspektes in den ANSI/X3/SPARC-Vorschlag besteht in der Berücksichtigung des Verteilungsaspektes auf jeder der Ebenen. Daraus resultieren verschiedene Architekturkonzepte für verteilte Datenbanken, die in unterschiedlichem Ausmaß die folgenden Kriterien erfüllen:

- Integrierung der lokalen Datenbasen zu einer globalen Datenbasis:
  Eine solche Integrierung ist aus zwei Gründen anzustreben. Einerseits soll den Benutzern eines verteilten Datenbanksystems eine Gesamtsicht der lokalen Datenbasen zur Verfügung stehen, die eine einfache, einheitliche Manipulation erlaubt. Andererseits soll die Kontrolle der Manipulationen an den Datenbeständen nicht nur isoliert auf die lokalen Datenbasen beschränkt, sondern auch auf deren wechselseitige Beziehungen ausgedehnt werden.
- Minimierung des Aufwandes für die Anpassung heterogener Datenbankschnittstellen:
  Die einheitliche Manipulation einer globalen Datenbasis und

die sich daraus ergebende Notwendigkeit zum Austausch von Daten zwischen Datenbasen verschiedener Rechner erfordert Mechanismen zur Konversion von Daten und Datenmanipulations-/Datendefinitionssprachen. In verteilten DV-Systemen mit Kleinrechnern sollten diese Konversionen auf ein Mindestmaß beschränkt werden (vgl. 2.7).

- Übernahme von konventionellen Datenbanktechniken:
  Die neue Technologie "Verteilte Datenbanken" sollte konsequent auf für konventionelle Datenbanksysteme bekannte und bewährte Architekturkonzepte, Datenmodelle und Zugriffsmethoden aufbauen. Im Idealfall kann dies die direkte Einbeziehung existierender konventioneller Datenbanksysteme in ein verteiltes Datenbanksystem bedeuten.
- Portabilität:
  Das verteilte Datenbanksystem muß auf verschiedene verteilte DV-Systeme und insbesondere auf Erweiterungen des zugrundeliegenden verteilten DV-Systems um zusätzliche Rechner übertragbar sein.

Die Architekturkonzepte für verteilte Datenbanksysteme werden im folgenden anhand dieser Kriterien diskutiert und bewertet. Im Einzelfall spielen bei der Abwägung des Für und Wider der Architekturkonzepte weitergehende, ebenenspezifische Kriterien eine Rolle, deren Bedeutung erst bei der Diskussion des Verteilungsaspektes auf den einzelnen Ebenen verständlich wird.

In den verschiedenen Architekturkonzepten werden die verteilungsspezifischen Aufgaben einer Schicht in einer einzigen Komponente, dem Verteiler, konzentriert. Die einzelnen Ebenen werden dadurch von Verteilungsaspekten abgeschirmt und damit verteilungsunabhängig. Der Verteiler wird als Hilfsmittel für die Diskussion eingeführt, bei der weniger die Verteilung von Funktionen der einzelnen Schichten als vielmehr die Verteilung von Daten, Transaktionen und Benutzern interessiert. Dem Verteiler wird - je nach Ansiedlung innerhalb des ANSI/X3/SPARC-Vorschlags - die Führung der entsprechenden Abbildungsinformationen und/die Durchführung der jeweiligen schichtenspezifischen Aufgaben zugeschlagen. Die Realisierung des Verteilers wird nicht behandelt. Formal lassen sich die ein-

zelnen Architekturkonzepte dadurch unterscheiden, zwischen welchen Ebenen der Verteiler in dem ANSI/X3/SPARC-Vorschlag angesiedelt wird. Die Ebenen oberhalb des Verteilers erhalten globale, die Ebenen unterhalb des Verteilers lokale Bedeutung.

Die Aufgaben des Verteilers hängen im einzelnen von dessen Ansiedlung innerhalb des ANSI/X3/SPARC-Vorschlags ab. Sie können zu sechs Aufgabenbereichen zusammengefaßt werden:

- Führung der Information zur Lokalisierung von Datenobjekten im verteilten DV-System (Netzkatalog),
- Verfügbarmachung der Daten aller Datenbasen im verteilten DV-System,
- Zwischenspeicherung von Daten,
- dynamische Verteilung von Transaktionen,
- koordinierte Überwachung der Transaktionsausführungen im verteilten DV-System,
- Konversion zwischen heterogenen lokalen Datenbankschnittstellen.

3.2 Verteilung auf der Ebene der externen Modelle

Es ergibt sich ein erstes Architekturkonzept für verteilte Datenbanksysteme, wenn man den Verteiler oberhalb der externen Modelle ansiedelt. Er verknüpft auf den verschiedenen Rechnern eines verteilten DV-Systems installierte konventionelle Datenbanksysteme.

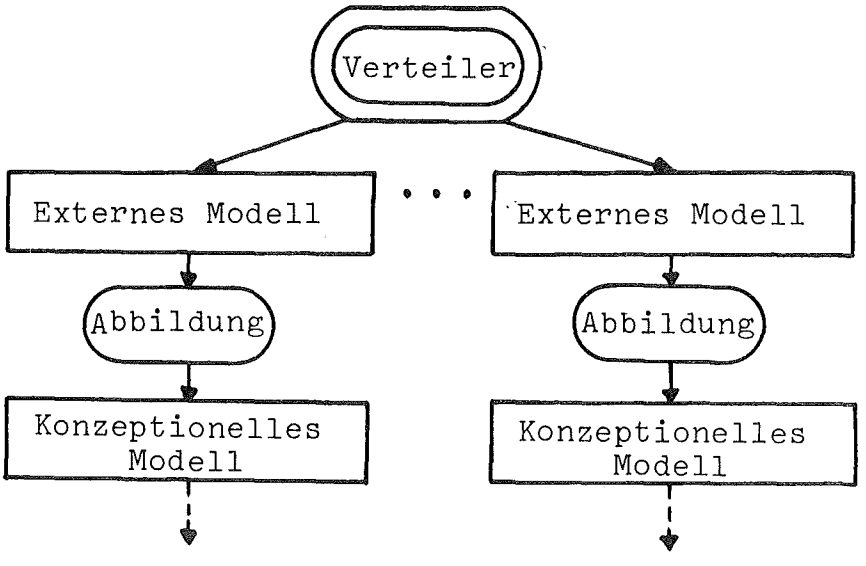


Bild 3.2: Realisierung eines verteilten Datenbanksystems durch Zusammenfassung aller Datenbasen auf der Ebene der externen Modelle

Die Vorzüge dieses Ansatzes liegen

- in der vollständigen Einbeziehung konventioneller Datenbanksysteme in ein verteiltes DV-System ohne Änderungen,
- in dem Zusammenschluß bereits existierender (installierter) Datenbanksysteme ohne Umorganisationen.

Dieser Ansatz liegt den meisten bisher in der Literatur diskutierten Vorstellungen über verteilte Datenbanksysteme zugrunde /CHA1, KAR1, LAC1, PEE1/.

Nachteilig wirkt sich bei diesem Architekturkonzept aus:

- Die Modellbildung ist auf allen Ebenen des verteilten Datenbanksystems lokal. Das hat z.B. zur Folge, daß Beziehungen zwischen Daten verschiedener Datenbasen nicht spezifizierbar sind: unkontrollierte Redundanzen zwischen verschiedenen Datenbasen können auftreten und zu Konsistenzverletzungen führen.
- Die Benutzerschnittstellen (Datenmanipulations- und Datendefinitionssprachen) verschiedener Datenbanksysteme sind i.a. nicht identisch. Umfangreiche Mechanismen zur Schnittstellenkonversion müssen vorgesehen werden. Dies kann insbesondere bei der Integration von existierenden Datenbanksystemen bei Erweiterung des verteilten DV-Systems Probleme aufwerfen und widerspricht damit der Forderung nach Portabilität.

Für dieses Architekturkonzept können in Abhängigkeit von der Realisierung der Benutzerschnittstelle zwei gegensätzliche Varianten unterschieden werden. Die erste Variante bürdet dem Benutzer sämtliche Aufgaben des Verteilers (vgl. 3.1) auf. Im Beispiel einer Anfrage, die sich über Datenbasen mehrerer Rechner erstreckt, umfassen diese:

- Aufspalten der Anfrage in Teilanfragen, die sich auf lokale Datenbanksysteme beziehen,
- Festlegen der Reihenfolge, in der die Teilanfragen auszuwerten sind,
- Konvertieren der Teilanfragen in die spezifischen lokalen Anfragesprachen,

- Zustellen der konvertierten Teilanfragen an die zuständigen lokalen Datenbanksysteme in der ermittelten Reihenfolge,
- Entgegennehmen und Zwischenspeichern der auf die einzelnen Teilanfragen zutreffenden Teilergebnisse,
- Konvertieren der Teilergebnisse in ein zur Weiterverarbeitung geeignetes Format,
- Zusammensetzen der Teilergebnisse zum Gesamtergebnis.

Diese erste Variante ist vergleichbar mit dem Multiple Systems Coupling (MSC)-Feature, mit dem Daten von verschiedenen IMS-Datenbasen abgerufen werden können /MCG1/.

Die zweite Variante schirmt den Benutzer eines verteilten Datenbanksystems von den verteilerspezifischen Aufgaben ab, d.h. der Verteiler wird in das verteilte Datenbanksystem integriert. Dies gelingt durch die Erweiterung des obigen Architekturkonzepts (Bild 3.2) um die Ebene "Netz-Modell"; auf dieser können Benutzersichten formuliert werden, die sich auch auf Datenbasen über mehrere Rechner erstrecken (Bild 3.3). Zu den einzelnen Datenbasen ist noch ein lokaler Zugang denkbar.

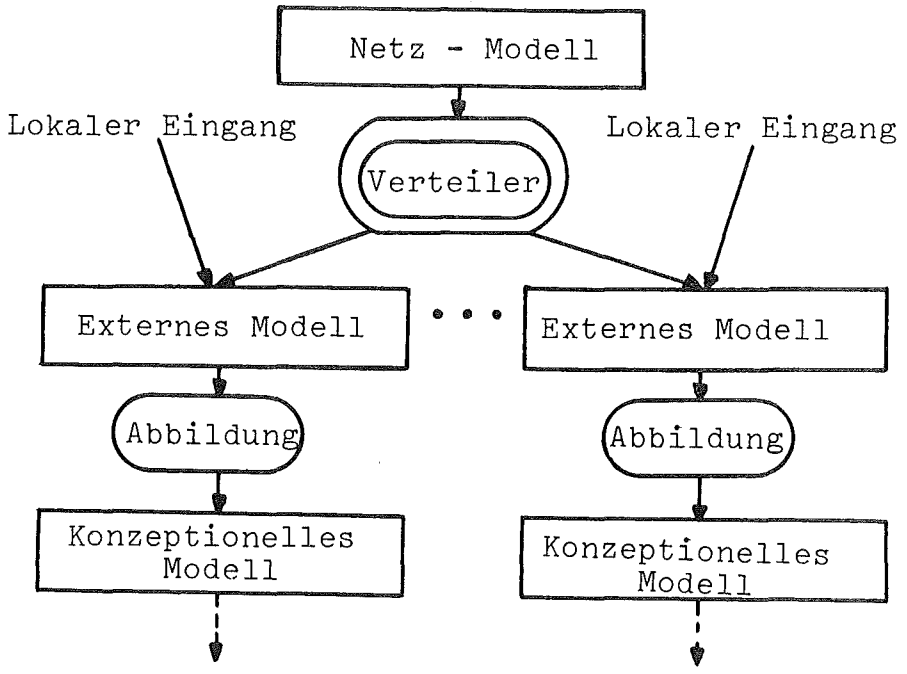


Bild 3.3: Realisierung eines verteilten Datenbanksystems durch Aufsetzen einer globalen Schnittstelle auf Datenbanksysteme

Ähnliche Ansätze, die auf der Einführung von Netz-Modellen beruhen, werden in /NBC1, SCH2, SCH3, SPT1/ beschrieben.

Es ist denkbar, auch für die Modellierung auf der Netz-Modell-Ebene jedem Benutzer sein Datenmodell zuzugestehen (Koexistenz-Ansatz /FAL1, NIJ1/).

Zur Umgehung der auf der mangelnden Integration aller Datenbasen beruhenden Nachteile werden in /GJP1/ Netz-Modelle mit Spezifizierungsmöglichkeiten für globale Integritätsbedingungen eingeführt, aus denen wiederum externe Modelle abgeleitet werden können. Eine solche Vorgehensweise erfordert redundante Systembausteine und ist daher für verteilte DV-Systeme mit Kleinrechnern nicht mit adäquatem Aufwand realisierbar.

3.3 Verteilung auf der Ebene der konzeptionellen Modelle

Bei der Anordnung des Verteilers zwischen den Ebenen der externen und konzeptionellen Modelle erhält man ein Architekturkonzept für verteilte Datenbanksysteme, in dem lokale Datenbasen, die aus der logischen Systemsicht voneinander isoliert sind, mittels globaler Benutzersichten in gewissem Ausmaß miteinander verknüpft werden können (Bild 3.4).

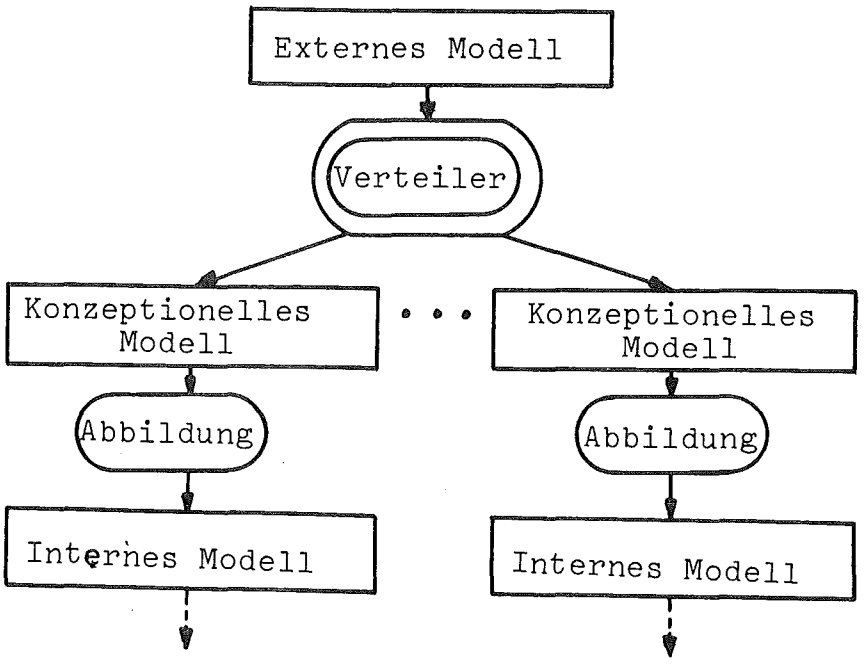


Bild 3.4: Realisierung eines verteilten Datenbanksystems durch Aufsetzen globaler Benutzersichten auf lokale logische Systemsichten

Dieses Architekturkonzept entspricht weitgehend dem in Bild 3.3 vorgestellten Konzept. Es unterscheidet im Gegensatz zu diesem nicht zwischen globalen und lokalen Benutzersichten. Der direkte Zusammenschluß konventioneller und existierender Datenbanksysteme in ein verteiltes Datenbanksystem ist nicht mehr ohne weiteres möglich.

3.4 Verteilung auf der Ebene der internen Modelle

Ordnet man den Verteiler zwischen den Ebenen der konzeptionellen und internen Modelle an, so erhält man ein Architekturkonzept für verteilte Datenbanksysteme, das sich von den bisher angeführten Konzepten insbesondere dadurch unterscheidet, daß eine vollständige Integrierung der Datenbasen aller Rechner zu einer globalen Datenbasis ermöglicht wird (Bild 3.5). Dies geschieht durch die Einführung eines globalen konzeptionellen Modells, das eine einheitliche Sicht aller Datenbasen gestattet.

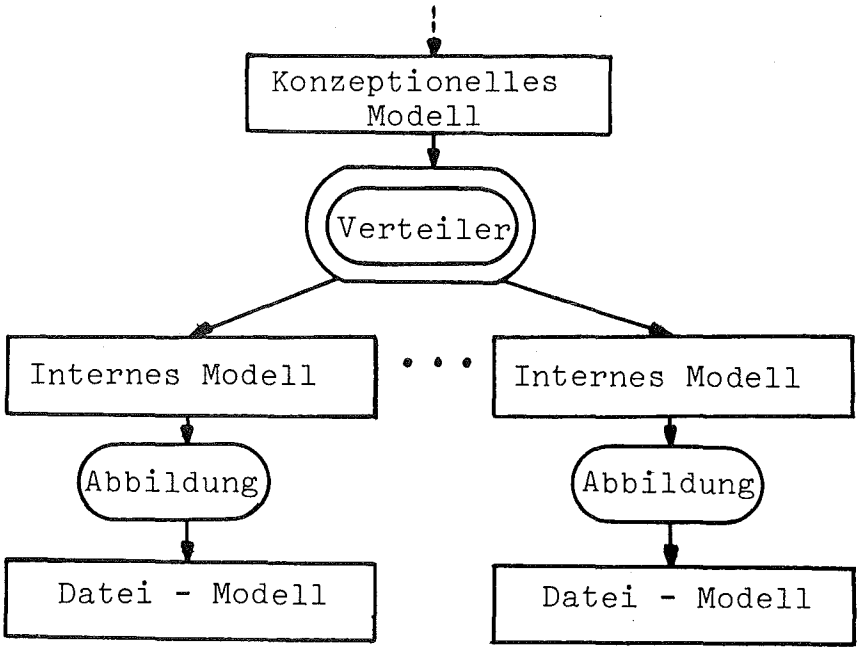


Bild 3.5: Realisierung eines verteilten Datenbanksystems durch vollständige Integration der Datenbasen auf der logischen Systemebene

Verknüpfungen zwischen Daten verschiedener Datenbasen, z.B. zur Spezifizierung globaler Integritäts- und Datenschutzbedingungen können formuliert, unkontrollierte Redundanzen zwischen Datenbasen unterschiedlicher Rechner vermieden werden. Die Verteilung der Daten kann - unter Berücksichtigung des Aufwands für die Einhaltung solcher Bedingungen - nun auch unter dem Aspekt der Minimierung dieses Aufwands vorgenommen werden. Anschaulich kann dies bedeuten, daß Daten, die durch Integritäts- und Datenschutzbedingungen verknüpft sind, möglichst auf einem Rechner abgelegt werden.

Für die Anpassung heterogener lokaler Datenbankschnittstellen ist in diesem Architekturkonzept ein gangbarer Weg vorgezeichnet. Ein globales konzeptionelles Modell kann als Standard aufgefaßt werden, an den die verschiedenen internen Modelle anzupassen sind. Auf diesem Standard kann auch der Datenaustausch zwischen den Datenbasen verschiedener Rechner beruhen.

Die Integrierung des Verteilers in konventionelle Datenbanksysteme ist bei diesem Architekturkonzept nicht mehr praktikabel. Dies liegt daran, daß in konventionellen Datenbanksystemen keine interne Schnittstelle identifizierbar ist, auf der der Verteiler aufgesetzt werden kann.

Zur Erhöhung der Portabilität kann eine globale Standardisierung der internen Modelle oder sogar der Datei-Modelle beitragen.

Von einem ähnlichen Architekturkonzept wird in /STN1/ ausgegangen. Als Datenmodell für die globale logische Systemebene wird eine um den Verteilungsaspekt erweiterte Variante des Relationenmodells /COD1/ vorgeschlagen.

3.5 Verteilung auf der Ebene der Datei-Modelle

Bei der Ansiedlung des Verteilers zwischen der Ebene der internen und der Datei-Modelle ergibt sich ein Architekturkonzept für verteilte Datenbanksysteme, das die Integrierung aller Datenbasen zu einer globalen Datenbasis auch auf Zugriffspfade ausdehnt (Bild 3.6).

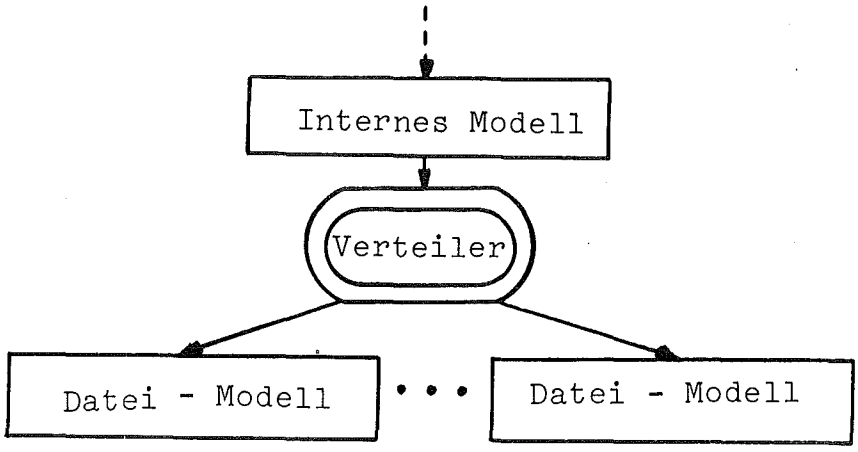


Bild 3.6: Realisierung eines verteilten Datenbanksystems mit Spezifikationsmöglichkeit für globale Zugriffspfade

Die Notwendigkeit zur Einführung globaler Zugriffspfade soll anhand eines Beispiels begründet werden: In einem relationalen Datenbanksystem selektieren verschiedene Benutzer an unterschiedlichen Rechnern Tupel derselben, beliebig verteilten Relation jeweils über verschiedene Attribute. Zur Zugriffsbeschleunigung seien für diese Attribute invertierte Dateien eingerichtet. Bei den bisher angeführten Architekturkonzepten ist die Ablage von invertierten Dateien an die Ablage der zugehörigen Relationen gekoppelt. Eine weitere Zugriffsoptimierung kann für bestimmte Fälle erreicht werden, wenn man diese Kopplung aufhebt, um z.B. die von den einzelnen Benutzern benötigten invertierten Dateien jeweils auf deren Anfragerechner abzulegen.

In dem vorliegenden Architekturkonzept können auch Zugriffspfade als Verteilungsobjekte betrachtet und dementsprechend beliebig verteilt werden. Für den allgemeinen Fall muß hierzu ein Konzept zur netzweiten Verzeigerung vorausgesetzt werden.

Zusammenfassend kann man feststellen, daß in diesem Architekturkonzept im Gegensatz zu den bisher diskutierten Konzepten eine Zugriffsoptimierung zusätzlich durch eine entsprechende Verteilung von Zugriffspfaden durchgeführt werden

kann.

Den restlichen Bewertungskriterien genügt auch dieses Architekturkonzept nicht (Argumentation ähnlich wie in 3.4).

Zur Erlangung zufriedenstellender Portabilität und Verwendbarkeit konventioneller Datenbanktechniken und zur Verringerung des Anpassungsaufwands für heterogene Datenbankschnittstellen bietet sich eine Alternative an, die auf der Einführung einer globalen Dateiverwaltungsschnittstelle - logisches Datei-Modell - beruht (Bild 3.7).

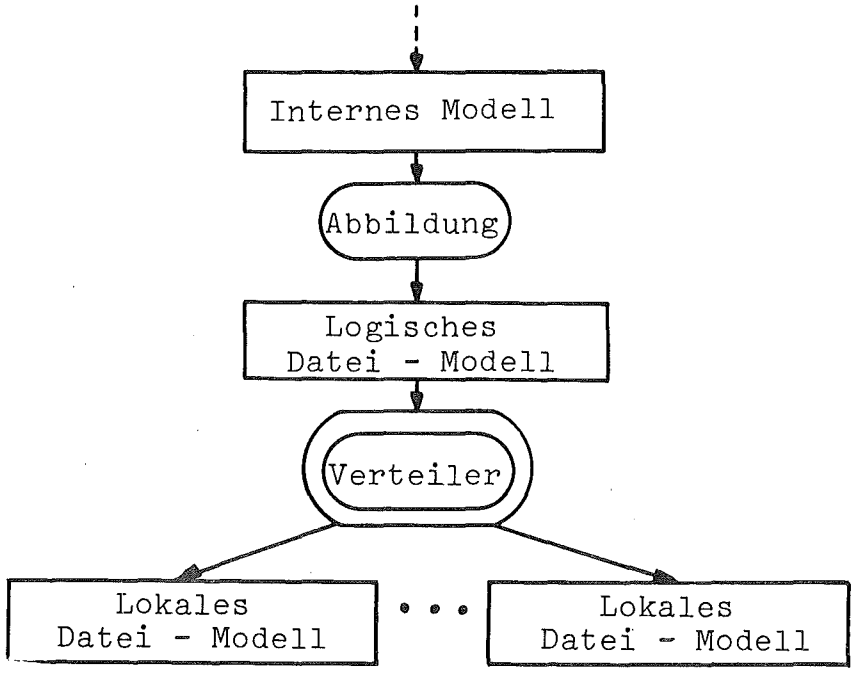


Bild 3.7: Realisierung eines verteilten Datenbanksystems auf der Basis eines verteilten Dateiverwaltungssystems

In diesem Architekturkonzept wird die Integrierung der Datenbasen aller Rechner bis auf Dateiebene fortgesetzt. Verteilungsobjekte sind Dateien und Sätze. Die verteilungsspezifischen Aufgaben werden vom Benutzer der globalen Dateiverwaltungsschnittstelle abgeschirmt.

Für eine Integrierung aller Datenbasen auf Dateiebene können zwei Gründe angeführt werden:

- In die Überlegungen bzgl. der Abspeicherung von Dateien lassen sich sämtliche im verteilten DV-System vorhandene Peripheriespeicher einbeziehen. Insbesondere können zur Optimierung der peripheren Zugriffe die in konventionellen

DV-Systemen gezielt steuerbare Beziehungen abgespeicherter Dateien (z.B. Dateien im selben Zylinder, Dateien auf verschiedenen Datenträgern) leicht auf eine verteilte Umgebung erweitert werden (z.B. Dateien auf Datenträgern verschiedener Rechner, Dateien auf Datenträgern benachbarter Rechner).

- Die Auflösung lokaler Überlaufsituationen durch ein Ausweichen auf andere Rechner erfordert eine globale Verwaltung der Peripheriespeicher, die nach dem ANSI/X3/SPARC-Vorschlag der Dateiebene zuzuschlagen ist.

Die Anpassung heterogener lokaler Datenbankschnittstellen wird reduziert auf die Anpassung der Dateiverwaltungen. Speziell bei einem auf Kleinrechnern basierenden verteilten DV-System ist der Aufwand für die erforderlichen Konversionen im Vergleich zu anderen Architekturkonzepten gering, da die meisten Kleinrechner mit Dateiverwaltungssystemen ähnlicher Leistungsfähigkeit ausgestattet sind.

Auf die globale Dateiverwaltungsschnittstelle können in konventioneller Technik gemäß dem ANSI/X3/SPARC-Vorschlag die übrigen Ebenen eines Datenbanksystems aufgesetzt werden. Geklärt werden müssen jedoch noch die Probleme bzgl. der Verteilung und des Zusammenspiels der funktionellen Komponenten der Schichten des Datenbanksystems und die Probleme bzgl. der Verteilung der Schemata, die die Beschreibungen der Ebenen und der Abbildungen enthalten.

Durch die standardisierte globale Dateiverwaltungsschnittstelle wird die Portabilität unterstützt.

3.6 Zusammenfassung

Jedes der in den vorstehenden Kapiteln vorgestellten Architekturkonzepte für verteilte Datenbanksysteme hat seine Berechtigung in Abhängigkeit von der Gewichtung der in 3.1 aufgestellten Bewertungskriterien. Verzichtet man z.B. auf die vollständige Integrierung aller Datenbasen zu einer globalen Datenbasis, so erfordert eine Realisierung des Architekturkonzeptes, das die Verteilungsaspekte auf der Ebene der externen Modelle ansiedelt (vgl. 3.2), den geringsten

Aufwand, da auf konventionelle Datenbanksysteme aufgebaut werden kann.

Ausgehend von den Bewertungskriterien wurden bei der Diskussion der einzelnen Architekturkonzepte folgende Forderungen an ein verteiltes Datenbanksystem aufgestellt und begründet:

- Die Spezifizierung der Benutzerwünsche soll mittels einer Datenmanipulations- und einer Datendefinitionssprache möglich sein.
- Es sollen Anfragen erlaubt sein, die sich auf Datenbasen in mehreren Rechnern beziehen.
- Die Verteilung der Daten soll für Benutzer nicht sichtbar sein.
- Eine globale logische Systemsicht soll mittels einer Datendefinitionssprache spezifizierbar sein.
- Globale Integritäts- und Datenschutzbedingungen sollen formuliert werden können.
- Es soll eine Zugriffsoptimierung bzgl. Benutzergruppen, die selbst über das verteilte DV-System verteilt sein können, ermöglicht werden.
- Die Verteilung von Zugriffspfaden soll unabhängig von der Verteilung von Primärdaten spezifiziert werden können.
- Es soll eine globale Verwaltung der Peripheriespeicher ermöglicht werden.

Aus der Sicht des ANSI/X3/SPARC-Vorschlags können diese Forderungen dadurch zusammengefaßt werden, daß für die Benutzersichten, die logische Systemsicht, die interne Systemsicht und die Datei-Sicht eine globale Modellierung verlangt wird.

Legt man diese Forderungen bei der Bewertung der verschiedenen Architekturkonzepte zugrunde, so werden diese nur von dem in Bild 3.7 vorgestellten Architekturkonzept erfüllt, welches auf einem verteilten Dateiverwaltungssystem basiert. Wie ein solches verteiltes Dateiverwaltungssystem für ein heterogenes verteiltes DV-System mit Kleinrechnern auszusehen hat, wird in Kapitel 4 und 5 diskutiert.

4. Verteiltes Dateiverwaltungssystem für verteilte Datenbanken

4.1 Logische Dateiebene

Die Eignung eines verteilten Dateiverwaltungssystems als Basis für verteilte Datenbanken hängt im wesentlichen von den Eigenschaften der angebotenen Schnittstelle - im folgenden logische Dateiebene genannt - ab.
Folgende Forderungen sind zu erfüllen:

- komfortable, einfache Operatoren,
- Mehrfachbenutzung der Dateien vom gesamten verteilten DV-System aus,
- Verteilungstransparenz.

Dateitypen:

Objekte auf der logischen Dateiebene sind zwei verschiedene Dateitypen:

- B-Dateien und
- Z-Dateien.

Eine B-Datei ist ein über einen symbolischen Namen beliebiger Länge im verteilten DV-System eindeutig identifizierbares Objekt. Sie kann beliebig viele bitstrukturierte Sätze fester Länge enthalten.

Eine B-Datei kann in Relation zu einer anderen logischen Datei beliebigen Typs plaziert werden (Quick-Option), um die Zugriffszeiten von abwechselnden Zugriffen zu derart gekoppelten Dateien günstig zu beeinflussen.

Die Sätze werden bei direktem Zugriff über vom System fortlaufend vergebene Satznummernidentifikationen referenziert. Bei sequentiellem Zugriff wird ein anonymer Name, der Cursor, als Referenz verwendet. Die Zugriffsorganisationen sind voneinander unabhängig und können ohne Einschränkungen nebeneinander verwendet werden.

Jeder Satz kann indirekt einer Benutzergruppe zugeordnet werden, indem er einer Gruppe von Sätzen, die ein benutzerorientiertes Cluster umfassen, zugewiesen wird. Der Sinn

dieser Benutzergruppenorientierung von Sätzen besteht darin, verschiedenen Benutzergruppen, die bestimmte Satzkontingente einer B-Datei besonders häufig referenzieren, das Privileg eines besonders schnellen Zugriffs zu derart zugeordneten, geclusterten Sätzen zu gewähren.

Diese Definition macht keine Einschränkungen über die Art der Verteilung einer B-Datei auf verschiedene Rechner.

Eine Z-Datei ist ebenso wie eine B-Datei eindeutig im gesamten verteilten DV-System identifizierbar. Sie kann beliebig viele Sätze variabler Länge umfassen, auf die direkt über Schlüssel zugegriffen werden kann. Dabei wird vorausgesetzt, daß der Schlüssel feste Länge besitzt und an fester Stelle relativ zum Satzanfang steht. Der Schlüssel definiert eine sequentielle Ordnung, wenn er als vorzeichenlose Dualzahl interpretiert wird. Sie wird für die sequentielle Zugriffsorganisation verwendet.

Eine Kopplung einer Z-Datei als Ganzes an Benutzergruppen ist möglich, und ebenso wie bei B-Dateien können Z-Dateien mit anderen Dateien in eine, die Zugriffszeit begünstigende Relation (Quick-Option) gesetzt werden.
Es wird festgelegt, daß eine Z-Datei als Ganzes auf jeweils genau einem Rechner zugänglich ist, d.h. nicht aufgeteilt werden darf.

Operatoren:

Die Operatoren zu den beiden Dateitypen (siehe Anhang A) sind in drei Klassen eingeteilt:

- Die DML (Data Manipulation Language) manipuliert, vernichtet, erzeugt, beschafft Objekte usw.,
- die DDL (Data Definition Language) definiert Schablonen für Objekte und deren Eigenschaften,
- die MDL (Mapping Definition Language) ermöglicht das Eingreifen eines Datenbasisadministrators (DBA) in den Abbildungsprozeß von Objekten der logischen Dateiebene auf die darunterliegende Ebene. Diese Klassifikation gilt entsprechend auch für die Operatoren der anderen Ebenen des Architekturmodells (Bild 4.1).

Gemeinsames Kennzeichen der DML- und DDL-Schnittstellen der logischen Dateiebene sind ihre Datenunabhängigkeit, insbesondere ihre Transparenz gegenüber Verteilung.

In der DDL werden von einem Benutzer nicht mehr Angaben verlangt, als er ohne Kenntnis des Systems und dessen Zustandes liefern kann, und die ausreichen, um die definierten Objekte bearbeiten zu können. Aus Kenntnis dieser Angaben ist es einem Datenbasisadministrator gegebenenfalls möglich, mit Hilfe der MDL die Abbildung auf die physikalische Umgebung zu steuern und zu beeinflussen.

Kern der DDL ist ein Definitionsoperator DEF, der eine Datei, ihren Namen, bei B-Dateien die Satzlänge, bei Z-Dateien den Schlüssel, Paßworte und gegebenenfalls eine Kopplung mit anderen Dateien definiert, sowie dem System eine Reihe von Hinweisen wie Abschätzungen des Dateiumfangs, Satzgruppencluster und deren Zuordnung zu Benutzergruppen mitteilt. Diese Hinweise werden katalogisiert und können vom System verwendet werden, um eine günstige automatische Abbildung auf die physikalische Speicherebene zu erzielen.

Die MDL besteht aus dem Abbildungsoperator MAP, der über entsprechende Parameter es dem DBA gestattet, alle systeminternen Plazierungsvorkehrungen, Redundanz- und Reorganisationsfragen zu steuern.

Den äußeren Rahmen bei Operationen auf Daten des verteilten Dateiverwaltungssystems bilden die Operatoren zur Transaktionsklammerung, BEGIN_TRANS und END_TRANS. Sie definieren Wiederaufsetzpunkte im Falle eines Systemzusammenbruchs und die Punkte, an denen die Integrität der Datenbasis vorausgesetzt wird. Nachdem die Dateien mit Hilfe von OPEN eröffnet worden sind, können die Transaktionen ihre Betriebsmittel, B- oder Z-Dateien bzw. einzelne Sätze von Dateien über die Sperroperation LOCK reservieren. Die Gefahr, daß bei sukzessiven Reservierungen innerhalb einer Transaktion Verklemmungen auftreten können, wird in Kauf genommen, um im Normalfall einen gegenüber einer globalen Transaktionsverwaltung verminderten Aufwand bei höherem Parallelitätsgrad

zur Geltung kommen zu lassen. Bei sukzessiven Reservierungen innerhalb einer Transaktion muß die Regel eingehalten werden, daß eine Freigabe eines Betriebsmittels innerhalb der Transaktion mittels UNLOCK noch nicht beantragt wurde (vgl. /ESW2/).

Den eigentlichen Kern der DML bilden die Funktionen zum Erzeugen (INSERT), Modifizieren (WRITE) und Beschaffen (READ) von Sätzen. Diese Funktionen können synchron oder asynchron ausgeführt werden. Bei asynchroner Ausführung besteht die Möglichkeit, mit Hilfe der Funktion WAIT explizit die normale Programmausführung mit dem Ein-/Ausgabevorgang zu synchronisieren durch Abwarten des Endes des Transfervorgangs. Weitere Operatoren dienen dem Erzeugen (CREATE) und Zerstören (DELETE) von Dateien unter Bezugnahme auf die durch die DDL definierten Schemata. Zusätzlich gibt es eine Funktion zum Beschaffen einer Satznummer (GETNAME) eines noch zu erzeugenden Satzes in Abhängigkeit von der Benutzergruppe des aufrufenden Benutzers.

4.2 Verwendbarkeit der logischen Dateiebene für unterschiedliche Datenmodelle

Gemäß dem von der ANSI/X3/SPARC-Gruppe /ANS1/ vorgeschlagenen Datenbankarchitekturkonzept könnte die Verwendbarkeit der logischen Dateiebene anhand eines internen Modells, eines konzeptionellen Modells oder anhand von externen Modellen geführt werden. Da heute einerseits weder allgemein anerkannte interne Modelle noch konzeptionelle Modelle existieren, andererseits aber im Bereich der externen Modelle eine gewisse Konsolidierung in dem Sinn festzustellen ist, als z.B. Relationenmodell und Netzwerkmodell gleichzeitig nebeneinander zur Modellierung der Benutzersichten zugelassen werden (Koexistenzansatz /FAL1, NIJ1/), wird die Verwendbarkeit der logischen Dateiebene für diese zwei Datenmodelle gezeigt. Es werden die Abbildungsmöglichkeiten der verschiedenen Datenstrukturen auf die Elemente der logischen Dateiebene untersucht, Optimierungsmöglichkeiten aufgezeigt und Realisierungsmöglichkeiten datenmodellspezifischer Zugriffspfade diskutiert.

Ausgangspunkt für die Diskussion sind folgende Optimierungsmöglichkeiten, die das verteilte Dateiverwaltungssystem anbietet:

- die benachbarte Ablage von Sätzen kann implizit über Satznummern oder über die Zuordnung zu Benutzergruppenclustern nur innerhalb von Dateien gesteuert werden,
- die benachbarte Ablage von Dateien wird über die QUICK-Option realisiert,
- die parallele Ablage von Sätzen/Dateien kann über die Zuordnung zu Benutzergruppenclustern erreicht werden,
- die kopierte Ablage von Sätzen/Dateien wird indirekt über den Kopiermechanismus für Benutzergruppencluster ermöglicht.

4.2.1 Relationenmodell

Eine Relation wird meist definiert als eine zeitlich veränderliche Teilmenge des kartesischen Produkts von Definitionsbereichen /COD2/. Sie kann als Tabelle dargestellt werden, deren Zeilen die Tupel und deren Spalten die Attribute der Relation ausmachen. Eine relationale Datenbasis beinhaltet eine Menge solcher Relationen.

Als Datenmanipulationssprache wird Relationenalgebra vorausgesetzt /COD3/, deren bedeutendste Operatoren PROJEKTION, konstanter VERBUND und variabler VERBUND sind.

Relationen können direkt mit B-Dateien implementiert werden, wobei Tupel durch Sätze identifiziert werden. Die Optimierung der Ablage der Relationen kann daran gemessen werden, in welchem Ausmaß sie eine effizientere Auswertung der relationenalgebraischen Operatoren bewirkt.

So unterstützen Tupelcluster, die über die verschiedenen Werte eines Attributes definiert werden, Operationen auf diesen Attributen (PROJEKTION, VERBUND, DIVISION). Tupelcluster sind über die benachbarte Ablage von Sätzen realisierbar.

Hybridclusterbildung von Tupeln /SCH5/ ist ein Mittel, um im Falle von 1:n-Beziehungen zwischen Relationen die Durch-

führung von variablen VERBUND-Operationen zu beschleunigen. Bei beliebigen n:m-Beziehungen kann der variable VERBUND durch zusätzliche Tupelverkettungstechniken unterstützt werden /SCB1/. Verkettungstechniken sind insbesondere dann anzuwenden, wenn das Vorkommen einer Relation an verschiedenen variablen VERBUND-Operationen zu berücksichtigen ist (Klassenbildung). Schließlich ist mit Split-Mechanismen eine effiziente Auswertung der PROJEKTION, aber auch, wie in /GOT1/ gezeigt, des VERBUND zu erreichen. Hybridclusterbildung, Verkettungstechniken und Split-Mechanismen werden von der logischen Dateiebene nicht direkt unterstützt. Hierfür muß bei Bedarf eine weitere Schicht aufgesetzt werden.

Über das Instrument der parallelen Ablage von Dateien bzw. Sätzen läßt sich eine parallele Ablage von Relationen so steuern, daß die Auswertung relationenalgebraischer Ausdrücke durch eine geeignete Aufspaltung in Teilausdrücke selbst parallelisiert werden kann /SMC1/. Der Kopiermechanismus erhält unter diesem Aspekt eine neue Berechtigung /GHO1/.

Als Zugriffspfade für Relationen sind invertierte Dateien und Pointer Arrays gebräuchlich /STO2, TSI1/. Während invertierte Dateien direkt mit den Z-Dateien der logischen Dateiebene aufgebaut werden können, müssen für Pointer Arrays zusätzliche Strukturierungsmittel geschaffen werden, die insbesondere deren variablem Charakter Rechnung tragen.

4.2.2 Netzwerkmodell

Das Netzwerkmodell geht zurück auf die Vorschläge der CODASYL Data Base Task Group, die in /COD1/ publiziert wurden. Das Kernstück dieses Datenmodells bildet der Satz (Record), dessen Typ durch eine beliebig komplexe Attributstruktur charakterisiert wird. Sätze können auf zweierlei Art aggregiert werden:

- Gebiete (Areas): enthalten disjunkte Satzmengen
- Mengen (Sets): spezifizieren 1:n-Beziehungen zwischen Ankersatztypen (Owner) und Bestandssatztypen (Member), wobei Hierarchie-

oder Netzbildung möglich ist.

Eine Datenbasis setzt sich aus einer Menge von Gebieten zusammen.

Für die Abbildung auf die logische Dateiebene bieten sich zwei Alternativen an:

(1) Direkte Abbildung von Gebieten auf B-Dateien

Infolge der festen Satzlänge von B-Dateien ist die Zuordnung von Gebieten auf B-Dateien so vorzunehmen, daß jedem Satztyp dieses Gebiets genau eine B-Datei entspricht. Als Optimierungsmöglichkeit ist hier die benachbarte Ablage aller zu einem Gebiet gehörigen B-Dateien vorgesehen.

Die Mengenbildung kann über explizite Verkettung der zusammengehörigen Anker- und Bestandssätze oder über Pointer Arrays realisiert werden. Für Verkettungstechniken und Pointer Arrays muß eine zusätzliche Schicht eingeführt werden.

Bei Hierarchien wird zur Optimierung der Mengenbildung im allgemeinen eine Clusterung der zu einem Ankersatz gehörigen Bestandssätze angestrebt. Eine Realisierung dieser Cluster ist über die benachbarte Ablage der entsprechenden Bestandssätze möglich. Mit der QUICK-Option können sogar Cluster aus Ankersätzen mit ihren zugehörigen Bestandssatzclustern unterstützt werden.

Bei Vernetzung kann eine eindeutige Clusterung nur über Redundanzen erreicht werden. Auch dafür eignet sich der Kopiermechanismus.

(2) Direkte Abbildung von Mengen auf B-Dateien

Eine Abbildung einer Menge auf genau eine B-Datei ist im allgemeinen Fall wegen der festen Satzlänge von B-Dateien nicht möglich. Insofern bleibt hier als einzige Möglichkeit, jedem Satztyp genau eine B-Datei zuzuordnen. Für die Realisierung der Mengenbildung mittels Verkettung oder Pointer Arrays gilt dann entsprechendes wie in (1).

Für Gebiete sind hier nur noch Clusterverfahren sinnvoll, die etwa auf der benachbarten Ablage der zugeordneten B-Dateien aufbauen könnten.

Als Zugriffspfade sind im Netzwerkmodell gebiets- bzw. mengenspezifische Indextabellen sowie Berechnungsverfahren vorgesehen. Als Basis für Indextabellen eignen sich die Z-Dateien. Berechnungsverfahren werden dagegen nicht direkt unterstützt.

4.2.3 Ergebnis

Trotz der nur groben Abschätzung ist in den vorstehenden zwei Kapiteln die Verwendbarkeit der logischen Dateiebene als Basis für relationale oder netzwerkartige Datenbanksysteme deutlich geworden; hierzu hat auch das Vorhandensein von Optimierungsmöglichkeiten beigetragen, die insbesondere die verteilte Umgebung berücksichtigen.

Mit dem geringeren zusätzlichen Aufwand ist bei der Konzipierung eines relationalen Datenbanksystems zu rechnen, insbesondere wenn auf Verkettungstechniken verzichtet und stattdessen eine strikte Trennung von Primär- und Sekundärdaten bevorzugt wird.
Verzichtet man weiterhin auf die direkte Unterstützung des variablen Verbunds durch über Pointer Arrays realisierte Zugriffspfade, die - wie in /STO2/ gezeigt -, überhaupt nur für ein sehr begrenztes Anfrageverhalten Vorteile bringt, und unterstützt stattdessen den variablen Verbund indirekt über invertierte Dateien, so müssen keine zusätzlichen Strukturierungsmittel mehr entwickelt werden. Diese Vorgehensweise trägt vor allem den durch Kleinrechner vorgegebenen Restriktionen Rechnung.

4.3 Architektur des verteilten Dateiverwaltungssystems

4.3.1 Konzept

Für das Architekturmodell bieten sich als Ausgangsbasis die innerhalb der Betriebssysteme der einzelnen Rechnersysteme realisierten Dateiverwaltungssysteme an.

Eine Reihe von Argumenten spricht für die Verwendung dieser lokalen Dateiverwaltungssysteme:

- Vereinfachte bzw. automatische Integration:
  Die Integration in die Betriebs- bzw. Programmiersysteme ist gegeben bzw. vereinfacht. Daraus resultiert ein bzgl. der Erstellung und Wartung des verteilten Dateiverwaltungssystems stark verminderter Aufwand. Insbesondere in Hinblick auf die Heterogenität gewinnt dieses Argument weiteres Gewicht.
- Keine Erstellung redundanter, funktional äquivalenter Systembausteine:
  Die von den Herstellern gelieferte Grundsoftware benötigt die Funktionen der mitgelieferten Dateiverwaltung. Der Nachteil von Doppelentwicklungen dürfte gegenüber dem Aspekt, zumindest zeitweise den Hauptspeicher mit doppelten Systembausteinen zu belegen und damit einen der größten Engpässe von Kleinrechnern weiter zu verengen, weniger schwer wiegen.
- Kompatibilität:
  Die Kompatibilität der Betriebssoftware von Rechnerfamilien überträgt sich auf das verteilte Dateiverwaltungssystem. Diese Eigenschaft ist besonders unter dem Gesichtspunkt der für Kleinrechnersysteme sich stürmisch entwickelten Hardwaretechnologien wertvoll.

Die Menge der Schnittstellen der verschiedenen, bereits vorhandenen lokalen Dateiverwaltungssysteme, die in den Rechnersystemen des Rechnernetzes vorzufinden sind, soll im folgenden L-Ebene (lokale Ebene) genannt werden.

Um die Portabilität möglichst vieler wesentlicher Komponenten des verteilten Dateiverwaltungssystems weiter zu verbessern, wird eine einheitliche, homogene Dateiverwaltungsschnittstelle über der L-Ebene, die G'-Ebene als Standard (Grundsystem-Dateiebene) eingeführt. An diese Schnittstelle müssen alle lokalen Dateiverwaltungssysteme der L-Ebene angepaßt werden.

Das Problem bei der Definition einer Standardschnittstelle besteht darin, einerseits die G'-Schnittstelle möglichst tief zu legen, um den portablen Teil des verteilten Dateiverwaltungssystems möglichst groß und den spezifischen Aufwand für die Anpassungen gering zu halten, und andererseits darin, nicht zu viele bestehende Funktionen komfortablerer Systeme außer acht zu lassen.

Aufbauend auf dem Standard der G'-Ebene erweisen sich verschiedene Schichten für eine modellhafte Realisierung der logischen Dateiebene als notwendig, um die zahlreichen Aspekte der Verteilung von Daten unterscheiden und klar herausarbeiten zu können.

Die G-Schicht (erweiterte Grundsystem-Datei-Schicht) wird eingeführt, um den auf einen lokalen Rechner begrenzten Namensraum für G'-Dateien auf das gesamte Rechnernetz zu erweitern. Die VG-Schicht (virtuelle G-Datei-Schicht) ermöglicht die Identifikation mehrerer Kopien einer G-Datei, während es die B-Schicht gestattet, eine B-Datei auch logisch aufzuteilen und Dateiüberlauf automatisch zu verhindern.

Bild 4.1 skizziert die Ebenen und Schichten, die im folgenden dann weiter präzisiert werden.

Die Schnittstellen der logischen Dateiebene sind bereits in 4.1 beschrieben, die G'-Ebene wird anschließend in 4.3.2 erläutert (Definitionen siehe Anhang A). Syntaktisch unterscheiden sich die restlichen zu beschreibenden Schnittstellen, die G- und VG-Ebene nur unwesentlich von der G'-Schnittstelle, kleinere Abweichungen werden bei der Diskussion der einzelnen zugehörigen Schichten deutlich.

Für die Abbildung von Dateien der logischen Dateiebene auf VG-Dateien bzw. von VG-Dateien auf G-Dateien usw. werden netzweit eindeutige Namensgebungsschemata benötigt, die durch den sog. B-Katalog bzw. VG-Katalog usw. realisiert werden und die jeweils in die entsprechende Schicht einzuordnen sind. Z.B. den G-Katalog kann man sich abstrakt als eine Relation mit G-Dateinamen als Schlüssel und G'-Datei-

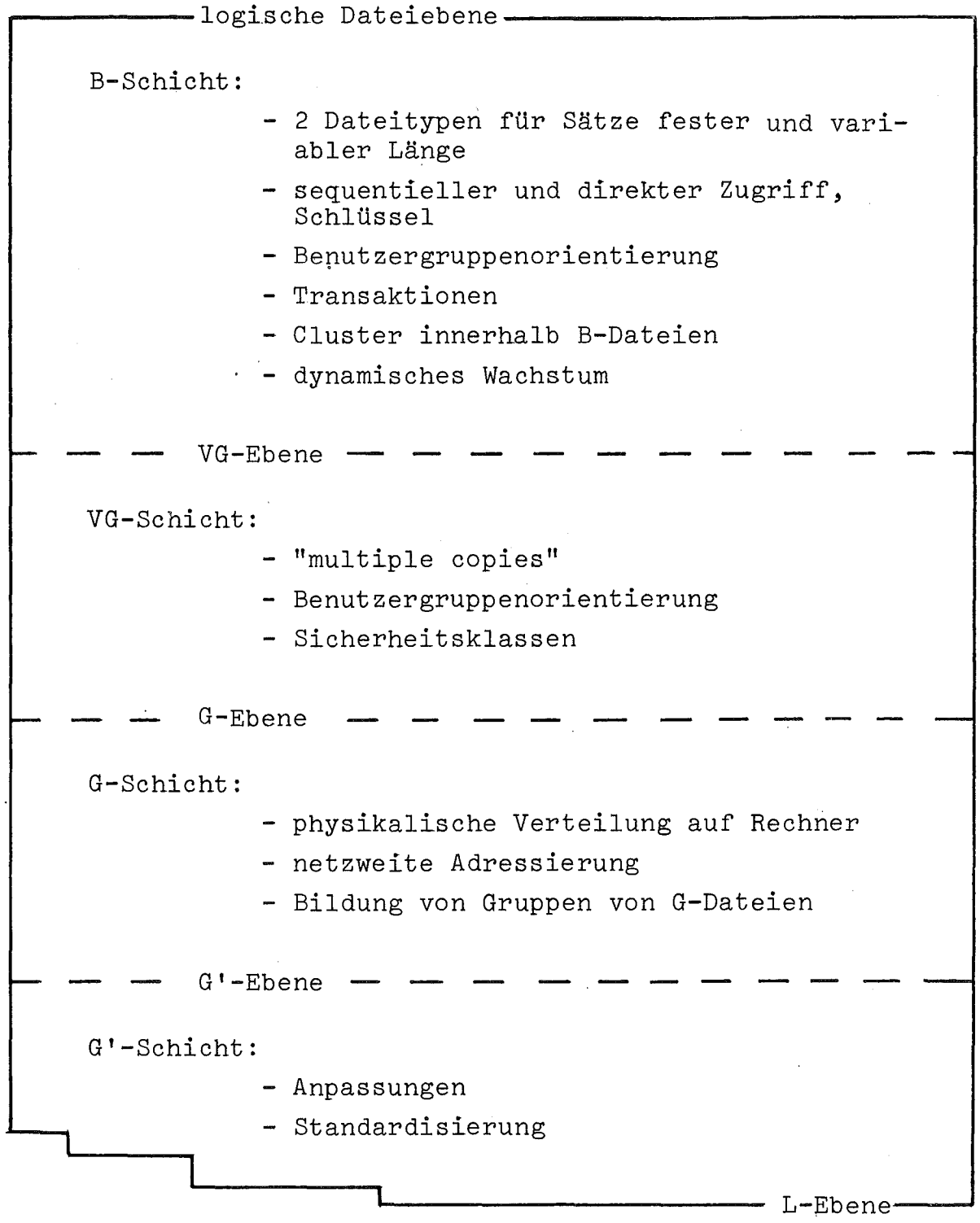


Bild 4.1: Schichtenstrukturierung des verteilten Dateiverwaltungssystems

namen als Attributwerten vorstellen. Daneben müssen für die einzelnen Dateien weitere Beschreibungsinformationen geführt werden.

Verschiedene Organisationsmöglichkeiten für diese Kataloge sind in 2.3 zu finden.

Das Ziel bei der Implementierung dieser Kataloge ist eine möglichst weitgehende Verwendung vorhandener Konzepte, d.h. insbesondere der angebotenen Dateitypen wie B- oder Z-Datei und die Vermeidung von Entwicklungen von speziell für Katalogisierungszwecke geschaffenen, völlig eigenständigen Organisationsformen, wie es häufig in herkömmlichen Betriebssystemen vorzufinden ist. Der Grund für diese Zielsetzung liegt darin, daß bei Kleinrechnern der Umfang des Systems eine besonders kritische Größe darstellt. Zusätzlich ergibt sich der Vorteil, daß der Implementierungs- aber auch der Wartungsaufwand für das Gesamtsystem verringert wird.

4.3.2 G'-Ebene

Die G'-Ebene definiert einen systeminternen Standard für alle lokalen Dateiverwaltungssysteme im verteilten DV-System. Sie bietet als einziges Objekt die rechnerweit eindeutig bezeichnete G'-Datei an.

Eine G'-Datei hat folgende Eigenschaften:

- sie besteht aus einer physisch kompakten Menge einer festen Anzahl von Blöcken,
- für alle G'-Dateien wird eine einheitliche Standardblocklänge verwendet,
- auf Blöcke kann lediglich direkt zugegriffen werden,
- Mehrbenutzerunterstützung
- Paßworte für Datenschutzzwecke
- Beeinflußbarkeit der Plazierung

Die Operatoren (siehe Anhang A) umfassen hauptsächlich Blocktransfer (READ, WRITE), die asynchron ausgeführt werden und deren Beendigung über WAIT abgewartet werden kann, Operatoren zum Eröffnen (OPEN), Schließen (CLOSE), Erzeugen

(CREATE), Zerstören (DELETE), Sperren (LOCK) und Freigeben (UNLOCK) von Dateien, der DEF-Operator definiert den Dateinamen, Dateiumfang, Paßworte und optional eine andere G'-Datei relativ zu der die Plazierung der zu definierenden Datei erfolgen soll.

Das mit der G'-Ebene festgelegte Abstraktionsniveau liegt damit vergleichsweise relativ nahe zur physikalischen Speicherebene von Direktzugriffsspeichergeräten, ist jedoch völlig geräteunabhängig.

4.3.3 G-Schicht

Aufgabe der G-Schicht ist die Realisierung eines für das gesamte verteilte DV-System geltenden rechnerunabhängigen Namensraumes.

Die G-Schnittstelle unterscheidet sich syntaktisch gegenüber der G'-Ebene in der DDL und DML nur unwesentlich. Die G-Datei hat im wesentlichen die gleichen Strukturen und Charakteristika wie G'-Dateien, jedoch kann die G-Ebene als eine auf das gesamte verteilte DV-System erweiterte physikalische Ebene angesehen werden.

Die MDL enthält außer einem MAP-Operator (vgl. MAP-Operator der logischen Dateiebene) zusätzlich einen COPY-Operator. Er gestattet es, anonyme Gruppen von inhaltsgleichen G-Dateien zu erzeugen, die alle denselben Umfang haben müssen. Diese Gruppen können aus Sicht der VG-Schicht ebenso wie einzelne G-Dateien als Objekte behandelt werden, auf die VG-Dateien (siehe 4.3.4) abgebildet werden können.

Als Randbedingung für den systeminternen Abbildungsalgorithmus der G-Schicht kann in der DDL der G-Ebene angegeben werden, welche G-Dateien nicht auf einem Rechner zusammen abgelegt werden sollen. Damit können Anforderungen an den Abbildungsalgorithmus, den Grad der Verfügbarkeit von mehreren G-Dateien möglichst hoch zu halten, entfallen. Aus der Sicht der Benutzer bzw. Planer des verteilten DV-Systems ist dies eine Voraussetzung dafür, die Verfügbarkeit des Gesamtsystems, eventuell unter Hinzunahme von Kopien (siehe

4.3.4), auch bei zeitweiligen Rechnerausfällen hoch auslegen zu können.

Die Abbildung einer G-Datei auf einen Rechner wird indirekt durch die Benutzergruppe, der die Datei bei ihrer Definition zugeordnet wurde, festgelegt. Es wird davon ausgegangen, daß die Aufteilung der Benutzergruppe auf eine Teilmenge der Rechner des verteilten DV-Systems invariant und dem System bekannt ist.

Die Quick-Spezifikation zwischen zwei G-Dateien kann in der Weise ausgewertet werden, daß sie auf ein und denselben Rechner oder benachbarten Rechnern abgelegt werden.

Der Abbildungsalgorithmus für die Abbildung von G- auf G'-Dateien kann vom Datenbasisadministrator außer Kraft gesetzt werden, indem er explizit mit Hilfe des MAP-Operators der G-Ebene die Zuordnung zu G'-Dateien vornimmt. Eine physisch benachbarte Ablage auf einem Datenträger oder eine parallele Ablage auf zwei simultan zugreifbaren Peripheriegeräten ist dagegen ausschließlich durch die G'-Schnittstelle beeinflußbar.

Die Einführung von Standards für Blocklänge und Dateiumfang für das gesamte verteilte Dateiverwaltungssystem erhöht die Kompatibilität und erleichtert Kommunikationsaufgaben. Der G-Dateiumfang kann ausschließlich ganzzahlige Vielfache bis zu einem Maximalwert eines bestimmten Grundkontingents von Blöcken annehmen. Der Wert für dieses Grundkontingent ist ein Generierungsparameter. Die Festsetzung von normierten Dateiumfängen bietet zudem bessere Voraussetzungen für eine optimale Ausnutzung der Peripheriespeicherkapazität auch unter Berücksichtigung, daß bei den einzelnen Rechnern Systemdateien benötigt werden, deren Umfang nicht den Standards für das verteilte Dateiverwaltungssystem genügt.

4.3.4 VG-Schicht

Aus der Sicht der G-Ebene sind zwei G-Dateien verschieden, wenn ihre Namen verschieden sind. Um inhaltsgleiche G-Dateien begrifflich zusammenzufassen, wird der abstrakte Typ VG-Datei (virtuelle G-Datei) eingeführt. Mehrere Kopien einer Datei, die zu einer Gruppe von G-Dateien (siehe 4.3.3) gehören, sind damit Inkarnationen einer VG-Datei. Die Aufgabe, mehrere Kopien konsistent zu halten - das Multi-Kopien-Problem (vgl. 2.8) - kann nun bei verteilungstransparenter DML und DDL auf der VG-Ebene der VG-Schicht übertragen werden.

Redundante VG-Dateien können entweder durch Bildung von Gruppen mit Hilfe der COPY-Anweisung der G-Ebene gebildet werden oder durch Angabe einer G-Dateiliste an Stelle eines Repräsentanten einer bereits bestehenden Gruppe direkt als Parameter des MAP der VG-Ebene. Durch das MAP werden damit quasi die G-Dateiliste bzw. die Gruppe mit einem VG-Dateinamen versehen.

Ebenso wie G-Dateien können VG-Dateien Benutzergruppen zugeordnet werden mit dem Vorteil, daß Benutzergruppen kürzere Zugriffszeiten auf die ihnen zugeordneten Dateien ermöglicht werden. Da die Verteilung der Benutzergruppen über das Rechnernetz als vergleichsweise invariant anzusehen ist, können kürzere Zugriffszeiten durch entsprechende Plazierung und gegebenenfalls durch Anlegen von Kopien erzielt werden. Falls eine Benutzergruppe auf verschiedene Rechner verteilt ist, müssen Anzahl und Plazierung von Kopien, durch die verbesserten Zugriffszeiten erreicht werden sollen, mit Hilfe einer lokalen Optimierung (vgl. 2.8) in Abhängigkeit von der relativen Häufigkeit von verändernden zu lesenden Dateizugriffen ermittelt werden. Aufgrund der oben genannten Eigenschaften ist die VG-Datei geeignet, um als Konstituent eines benutzerorientierten Clusters (siehe 4.1) von Sätzen herangezogen zu werden.

Durch Erhöhen des Redundanzgrades lassen sich Sicherheits- und Verfügbarkeitsanforderungen unterstützen. Hierzu ist vorgesehen, eine VG-Datei bei ihrer Definition mit unterschiedlichen Sicherheitsgraden zu versehen, und damit in-

direkt die Anzahl ihrer Inkarnationen festzulegen. Für diese Inkarnationen ist in der DDL der G-Schnittstelle der Wunsch spezifizierbar, daß eine G-Datei A nicht mit einer anderen G-Datei B auf ein und denselben Rechner untergebracht werden darf.

Während zur Erhöhung des Verfügbarkeitsgrades einfache Lösungen zur Bestimmung einer passenden Anzahl von Kopien herangezogen werden können, können die in der Literatur vorgeschlagenen Verfahren zur Zugriffszeitoptimierung (vgl. 2.8) wegen ihres Aufwands nur über separate Dienstprogramme implementiert werden, wobei dann der Datenbasisadministrator für das Bereitstellen der entsprechenden Kopien zu sorgen hat.

4.3.5 B-Schicht

Die in dieser Schicht anfallenden Hauptaufgaben sind das Abbilden von sowohl B- als auch Z-Dateien auf VG-Dateien, das Abbilden von Sätzen auf Blöcke sowie die Verwaltung von Transaktionen. Weiterhin ist die Beziehung zwischen benutzerorientierten Clustern von Sätzen, Blöcken, B- und VG-Datei festzulegen.

Die Sätze einer B-Datei werden entsprechend der lückenlosen Satznummern physisch fortlaufend in den Blöcken von VG-Dateien, aus denen eine B-Datei zusammengesetzt wird, abgelegt. Die Realisierung dieser Abbildung muß i.a. über eine Montage von Sätzen zu Blöcken bzw. Blöcken zu Sätzen, je nach Transferrichtung erfolgen, denn als Blockungsfaktoren sind rationale Zahlen zugelassen. Die fortlaufende Ablage der Sätze macht Abbildungstabellen entbehrlich und ermöglicht deshalb günstigere Zugriffszeiten, ohne zusätzlich Hauptspeicher zu beanspruchen. Der letztere Aspekt erweist sich besonders in Hinblick auf die Kleinrechnerumgebung von Vorteil.

Um eine einfache Abbildung von Sätzen auf Blöcke und auf VG-Dateien zu ermöglichen, wird eine B-Datei auf verschiedene, jedoch gleichlange VG-Dateien abgebildet. Pro VG-Datei wird eine ganzzahlige Anzahl von Sätzen untergebracht und somit

VG-Dateien überspannende Sätze vermieden. Der Umfang der VG-Dateien wird aus einer bei Definition der B-Datei angegebenen Umfangabschätzung zusammen mit Angaben über Clusterumfänge abgeleitet.

Benutzergruppen zugeordnete Cluster können optimal auf die gleichlangen VG-Dateien abgebildet werden, indem die Cluster auf Sätze mit dicht zusammenliegenden Satznummern beschränkt werden. Die Cluster werden, wenn möglich, in einer einzelnen oder mehreren benachbarten VG-Dateien untergebracht. Bei den gegebenen Voraussetzungen sind deshalb Satznummernintervalle festen Umfangs, definiert durch eine Anzahl eines Grundkontingents von Sätzen, zu wählen.

Die Forderung nach gleichem Umfang der VG-Dateien, die eine bestimmte B-Datei bilden, kann bewirken, daß bei sehr unterschiedlichen Clusterumfängen ein größeres Cluster auf mehrere VG-Dateien abzubilden ist. Dieser Nachteil kann jedoch durch eine Quick-Spezifikation der VG-Dateien untereinander wieder weitgehend beseitigt werden. Die Beschränkung auf statische Clustergröße macht dynamische Reorganisationen bzw. erhöhten Speicherbedarf entbehrlich. Allerdings kann man letzeres auch durch großzügige Abschätzung der Clustergröße erreichen, wobei dann dynamische Anpassungen nur selten noch Gewinne bringen gegenüber nach längeren Zeiträumen stattfindenden Reorganisationen durch den Datenbasisadministrator.

Der Nachteil für einen Benutzer, ein festes Satzintervall zu verwenden, erscheint gegenüber der Notwendigkeit, die Grenzen dieses Intervalls berücksichtigen zu müssen, wegen der ohnehin relativ künstlichen Satznummernidentifikationen weniger einschneidend zu sein. Das Erfordernis hingegen, daß verschiedene Cluster disjunkt sein müssen, obwohl ein Cluster auch mehreren Benutzergruppen zugeordnet sein kann, ist für die Kleinrechnerumgebung als sinnvoll zu erachten.

Das folgende Beispiel zeigt, wie Benutzergruppen und Cluster verwendet werden können.

Eine Bank habe in einer Stadt eine Hauptzweigstelle und zwei Filialen. Die Hauptzweigstelle berechnet Zinsen und erstellt

Statistiken über Kontobewegungen, Kunden etc., die Filialen sind nur für das laufende Geschäft direkt mit dem Kunden zuständig. Sie führen die Konten der ihnen zugeordneten Kundenkreise uns erteilen Auskünfte über sich nur in Tageszeiträumen verändernde Informationen wie Aktien- oder Wechselkurse. Die beiden Filialen bilden zwei verschiedene Benutzergruppen, da ihre Kompetenzen nur für eigene Konten gelten, obwohl alle Konten in einer einzigen B-Datei geführt werden und obwohl die Anwendungen die gleichen sind. Die Benutzer in der Hauptzweigstelle, die speziell für Zinsberechnung und Statistiken zuständig sind, können beiden Benutzergruppen zugleich angehören oder sie bilden, bezogen auf die Kontendatei, eine eigene Benutzergruppe. Ihre Anwendungen sind verschieden, sie benötigen jedoch die gleichen Daten wie ihre Filialen. Treten innerhalb der Daten einer Benutzergruppe selbst wieder Cluster, z.B. die Konten der "Großkunden" einer Filiale auf, so können dieser Filiale (= Benutzergruppe) mehrere Cluster zugeordnet werden. Die Informationen über Aktienkurse können in einer B-Datei mit drei Inkarnationen, d.h. bei allen Filialen existieren und abends von der Hauptzweigstelle auf den neuesten Stand gebracht werden.

Man erkennt, daß Benutzergruppen nicht notwendig disjunkt sein müssen, obwohl die zugehörigen Satzcluster als disjunkt oder als identisch vorausgesetzt werden.

Eine dynamische Komponente bei der Abbildung einer B- oder Z-Datei auf VG-Dateien liegt in der Aufgabe, gegebenenfalls Dateiüberlauf zu verhindern durch Hinzunahme einer weiteren VG-Datei oder, falls nötig, VG-Dateien freizugeben, wenn im entsprechenden Satznummernintervall alle Sätze beseitigt (REMOVE) worden sind.

Für Z-Dateien ist eine andere Abbildung von Sätzen auf Blöcke notwendig wegen der variablen Satzlänge und des Zugriffs über Schlüssel. Die Beschränkung, eine Z-Datei auf genau einem Rechner unterzubringen, ermöglicht Verzeigerungen, ohne Ineffizienzen und Verfügbarkeitsfragen bei Verzeigerungen über verschiedene Rechner berücksichtigen zu müssen.

Die variable Satzlänge und die nicht nach oben festgelegte Anzahl von Sätzen einer Z-Datei erfordern i.a. eine dynamische Abbildung auf mehrere VG-Dateien, die untereinander mit Hilfe der Quick-Spezifikation der VG-Ebene gekoppelt sind. Für sequentielle und direkte Zugriffsorganisationen bieten sich Baumorganisationen, z.B. nach /BAM1, HAE1/ an.

Die Zuordnung einer Z-Datei zu Benutzergruppen erfolgt so, daß die entsprechenden VG-Dateien wie in 4.3.4 beschrieben, zugeordnet werden.

5. Operationales Modell für ein verteiltes Dateiverwaltungssystem

5.1 Funktionen

Nachdem in Kap. 3 und 4 ausschließlich die Verteilung von Daten behandelt wurde, wird im folgenden der Aspekt der Verteilung von Funktionen näher untersucht.

Der Zugriff von Benutzern auf Daten erfolgt durch Absetzen von Operatoren der logischen Dateiebene an das verteilte Dateiverwaltungssystem. Zur Bearbeitung dieser Operatoren durch das verteilte Dateiverwaltungssystem müssen i.a. mehrere Rechner kooperieren, um neben der Durchführung von Datentransfers parallele Zugriffe mehrerer Benutzer zu regeln.

In einem operationalen Modell eines verteilten Dateiverwaltungssystems sind daher Funktionen bereitzustellen, die folgende Aufgaben zu lösen haben:

- Interpretation der Operatoren der logischen Dateiebene, aus denen die Benutzeraufträge bestehen,
- Transformation der Operatoren der logischen Dateiebene auf Anweisungen, die von den jeweiligen Rechnern bearbeitet werden können,
- Bereitstellung und Verwaltung von Information zur Durchführung von Zugriffen auf Primär- und Sekundärdaten,
- Verwaltung der Benutzeraufträge,
- Kontrolle der Zugriffe in einer Mehrbenutzerumgebung durch Transaktionsverwaltung,
- Übernahme von Datensicherungs- und Datenschutzmaßnahmen.

Datenschutz ist bisher auf ein Überprüfen von Paßwörtern beschränkt; Datensicherung wird im folgenden nur soweit betrachtet, als dies ihre Einbeziehung in Mechanismen zur Sicherung der operationalen Integrität (s. Kap. 6) erfordert.

5.2 Aufteilung in funktionelle Komponenten

Die Erledigung der Aufgaben des verteilten Dateiverwaltungssystems ist funktionellen Komponenten zu übertragen. Bei der Definition dieser funktionellen Komponenten ist zu berück-

sichtigen:

- das in Kap. 4 vorgestellte Architekturkonzept,
- die Systemeffizienz.

Die Systemeffizienz ist beeinflußbar durch die Strukturierung des Gesamtsystems. Es lassen sich folgende Forderungen an ein verteiltes Dateiverwaltungssystem stellen:

- Möglichkeit der Umkonfigurierung durch Änderung der Zuordnung von funktionellen Komponenten zu Rechnern,
- Austausch von Verfahren, z.B. Einführung anderer Sperrmechanismen,
- Optimierung der Verarbeitung paralleler Zugriffe mehrerer Benutzer.

Einen Ansatz zur Konzipierung des operationalen Modells für ein verteiltes Dateiverwaltungssystem stellt die explizite Implementierung jeder Schicht des in Kap. 4 vorgestellten Architekturkonzepts dar. Diese Vorgehensweise birgt jedoch Nachteile in sich:

Sie ist sehr aufwendig und effizienzmindernd durch die Ansiedlung gleichartiger Funktionen in verschiedenen Schichten, z.B. für die Katalogverwaltung.

Diese Überlegungen führen dazu, daß man einzelnen funktionellen Komponenten Aufgaben mehrerer Schichten überträgt und Information über die Abbildung zwischen den Ebenen des Architekturkonzepts zusammengefaßt wird (z.B. Kataloginformation).

Es ist denkbar, das operationale Modell eines verteilten Dateiverwaltungssystems dergestalt zu konzipieren, daß eine einzige funktionelle Komponente alle Funktionen erfüllt und diese Komponente auf jedem Rechner realisiert wird. Bei diesem Konzept ist jedoch die Berücksichtigung der unterschiedlichen Leistungsfähigkeit der Rechner in einem heterogenen Rechnernetz ausgeschlossen. Dies steht der Forderung nach Umkonfigurierbarkeit entgegen. Um dieser Forderung ebenfalls zu genügen, ist eine weitergehende Aufteilung in mehrere funktionelle Komponenten erforderlich.

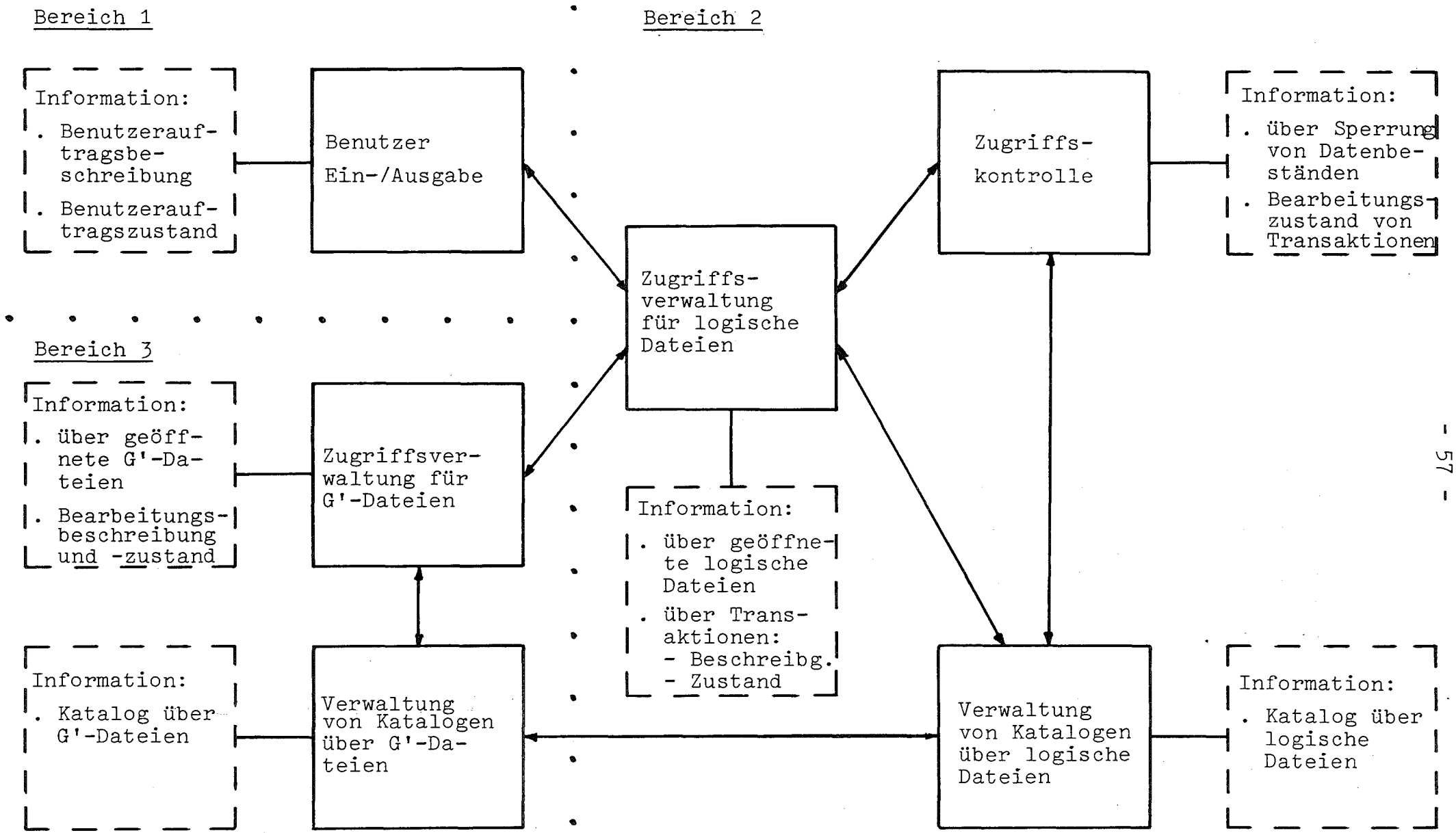


Bild 5.1: Übersicht über das operationale Modell eines verteilten Dateiverwaltungssystems

Benutzer Ein-/Ausgabe

Die Aufträge der Benutzer mit Anweisungen der logischen Dateiebene werden entgegengenommen und an die Zugriffsverwaltung für logische Dateien weitergeleitet. Die Benutzer Ein-/Ausgabe Komponente ist ebenfalls für die Übergabe der Bearbeitungsresultate an den Benutzer zuständig. Auf Anordnung der Zugriffsverwaltung für logische Dateien führt sie die Rücksetzung von Transaktionen durch. Die funktionelle Komponente muß für jeden ihrer Benutzeraufträge Auftragsbeschreibung, Zustandsbeschreibung und Rücksetzinformation führen.

Zugriffsverwaltung für logische Dateien

Hier werden die Benutzeranweisungen analysiert und interpretiert. Anschließend werden die Anweisungen in Zugriffe auf G'-Dateien transformiert und der Zugriffsverwaltung für G'-Dateien übergeben. Die Transformation basiert auf Information zur Lokalisierung, Erweiterung und Reduzierung von Datenbeständen, die von der Verwaltung von Katalogen über logische Dateien geliefert wird. Zugriffe auf logische Dateien sind nur innerhalb von Transaktionen erlaubt. Der Aufbau einer Transaktion hat bestimmten Regeln zu genügen, deren Einhaltung überprüft werden muß. Vor der eigentlichen Durchführung des Zugriffs, d.h. vor dem Absetzen der transferierten Anweisungen an die Zugriffsverwaltung für G'-Dateien, ist durch Übergabe von Sperranweisungen an die Zugriffskontrolle deren Zustimmung einzuholen. Zur Verwaltung der Transaktionen ist Information zu führen über:
Transaktionsidentifikation, Transaktionspriorität, die für diese Transaktion gesperrten Datenbereiche und der Bearbeitungszustand der Transaktion.

Verwaltung von Katalogen über logische Dateien

Die Aufgaben dieser funktionellen Komponente sind:

- Bereitstellung der Information über die Abbildung zwischen logischen Dateien und G'-Dateien für die Zugriffsverwaltung für logische Dateien,

- Eintragen/Streichen von Dateinamen bei Hinzufügung/Eliminierung von Dateien (logische Dateien, VG-Dateien und G-Dateien),
- Gewährleistung netzweit eindeutiger Dateinamen (logische Dateien, VG-Dateien und G-Dateien),
- Erzeugen/Löschen von Dateien.

Zur Erfüllung dieser Aufgaben kommuniziert die funktionelle Komponente mit der Verwaltung von Katalogen über G'-Dateien und mit der Zugriffsverwaltung für logische Dateien.

Zugriffskontrolle

Die Aufgabe der Zugriffskontrolle ist die Sicherung der operationalen Integrität von Primär- und Sekundärinformation. Die Zugriffskontrolle ist verantwortlich für Sperren/Freigeben von Datenbeständen und die Behandlung von Verklemmungssituationen. Wegen Verfolgung der Strategie des Zulassens von Verklemmungen müssen diese erkannt und beseitigt werden, wobei unter Umständen die Rücksetzung von Transaktionen erforderlich ist. Die Zugriffskontrolle hat Information zu unterhalten über die Sperrung von Datenbeständen sowie den Zustand der Bearbeitung von Transaktionen. Die Problematik der Sicherung der operationalen Integrität wird in Kap. 6 näher behandelt.

Zugriffsverwaltung für G'-Dateien

Diese funktionelle Komponente erhält Anweisungen über Zugriffe auf G'-Dateien von der Zugriffsverwaltung für logische Dateien. Sie analysiert und interpretiert die Anweisungen und transformiert sie auf Operatoren der L-Ebene der lokalen Dateiverwaltungssysteme. Die für die Transformation notwendige Abbildungsinformation liefert die Verwaltung von Katalogen über G'-Dateien.

Verwaltung von Katalogen über G'-Dateien

Die Aufgaben der Komponente umfassen:

- Bereitstellen der Information über die Abbildung von G'-Dateien auf L-Dateien für die Zugriffsverwaltung für G'-

Dateien,

- Eintragen/Streichen von Dateinamen bei Hinzufügung/Eliminierung von Dateien (G'-Dateien, L-Dateien),
- Erzeugen/Löschen von Dateien (G'-Dateien, L-Dateien).

Auf Anforderung der Verwaltung von Katalogen über logische Dateien stellt sie G'-Dateien zur Verfügung, auf Anforderung der Zugriffsverwaltung für G'-Dateien wird die Abbildungsinformation von G'-Dateien auf L-Dateien übermittelt.

5.4 Realisierung des operationalen Modells

Ein Exemplar einer funktionellen Komponente in einem Rechner heißt Funktionseinheit. Die Verteilung von funktionellen Komponenten geschieht über die Ansiedlung von Funktionseinheiten im verteilten System.

Die funktionellen Komponenten der Bereiche 1 und 3 müssen in jeweils einer Funktionseinheit auf jedem Rechner vorliegen, auf dem Benutzer bzw. Daten des verteilten Dateisystems vertreten sind.

Die funktionellen Komponenten des Bereichs 2 müssen jeweils in mindestens einem Exemplar im verteilten Dateiverwaltungssystem existieren. Mittels Verteilung dieser Funktionseinheiten läßt sich für ein gegebenes verteiltes DV-System die geeignete Struktur des verteilten Dateiverwaltungssystems konfigurieren; dies ist insbesondere für heterogene Rechnernetze von Bedeutung. Durch die Verteilung wird ein Spektrum von Strukturen angeboten, das als Extrema enthält:

- eine dezentrale Struktur, in der jede funktionelle Komponente auf jedem Rechner vertreten ist,
- zentrale Strukturen, in denen jede funktionelle Komponente des Bereichs 2 in genau einem Rechner (nicht notwendigerweise derselbe Rechner) existiert.

Durch die Verteilung der funktionellen Komponenten entstehen folgende Probleme:

- Konsistenzhaltung der von den Funktionseinheiten geführten

Information.

Dies ist durch Synchronisation der Zusammenarbeit der involvierten Funktionseinheiten zu lösen; einen Ansatz dazu bieten die Mechanismen der Zugriffskontrolle.

- Bestimmung der optimalen Anzahl der Exemplare einer funktionellen Komponente des Bereichs 2 und deren Plazierung im Netz.

  Zur Lösung dieses Problems sind Optimierungsmodelle zu entwickeln (vgl. 2.8).

- Zuordnungen zwischen Funktionseinheiten unterschiedlicher funktioneller Komponenten.

  Es sind mehrere Alternativen denkbar, z.B.:

  1.) Vorgabe einer statischen Zuordnung.
  2.) Jede Funktionseinheit wählt aufgrund eines vorgegebenen Algorithmus die anzusprechenden Funktionseinheiten aus.
  3.) Die Zuordnung wird durch eine funktionelle Komponente vorgenommen, die nicht dem verteilten Dateiverwaltungssystem angehört (z.B. Nachrichtentransportsystem) oder neu in dasselbe aufgenommen werden müßte.

Die beiden letztgenannten Aufgaben, Plazierung und Zuordnung, sind zweckmäßigerweise in Zusammenarbeit mit dem Datenbasisadministrator zu lösen, da die Verteilung der Funktionseinheiten des verteilten Dateiverwaltungssystems nicht unabhängig von der Verteilung der Daten ist.

6. Dezentralisierte Zugriffskontrolle in Mehrbenutzerumgebungen

6.1 Strukturierung der Problematik

Die integrierte Bearbeitung von Benutzeraufträgen durch das verteilte Dateiverwaltungssystem erfordert Mechanismen zur Koordinierung der Aktivitäten der Funktionseinheiten. Der Aufbau der Koordinierungsmechanismen richtet sich nach der Art der zu koordinierenden Aktivitäten und der zu beteiligenden Funktionseinheiten.

Um grundlegend die Konstruktion von Koordinierungsmechanismen zu untersuchen, wurde als konkreter Bezug die Kooperation zwischen Funktionseinheiten der Zugriffskontrolle ausgewählt, da in diesem Bereich bei dezentraler Kontrollstruktur und Mehrbenutzerumgebung Koordinierungsprobleme unterschiedlichster Natur existieren.

Eine Funktionseinheit der Zugriffskontrolle heißt im folgenden kurz Kontrolleinheit. Aus Vereinfachungsgründen wird vorausgesetzt, daß in jedem Rechner des verteilten DV-Systems eine Kontrolleinheit eingerichtet ist, der die lokal vorhandenen Datenbestände zur Zuweisung an Transaktionen zugeordnet sind (vgl. Bild 6.1).

Die Kontrolleinheiten müssen kooperieren

- zur parallelen Bearbeitung unterschiedlicher Transaktionen unter Sicherung der operationalen Integrität der Datenbestände bei störungsfreiem System,
- zur Unterstützung der vom Gesamtsystem zu ergreifenden Maßnahmen bei Ausfall und Wiedereingliederung von Datenbeständen und Kontrolleinheiten.

Aus diesen Aufgabenbereichen ergeben sich Koordinationsanforderungen als Aufträge an die Kontrolleinheiten, bestimmte Aktivitäten koordiniert durchzuführen. Koordinationsanforderungen sowie Nachrichten von anderen Funktionseinheiten werden in Warteschlangen geführt. Jede Kontrolleinheit basiert ihre Entscheidungen auf Information über den Zustand von Kontrolleinheiten, Datenbeständen, Transaktionen und den restlichen Funktionseinheiten, mit denen sie kommuniziert.

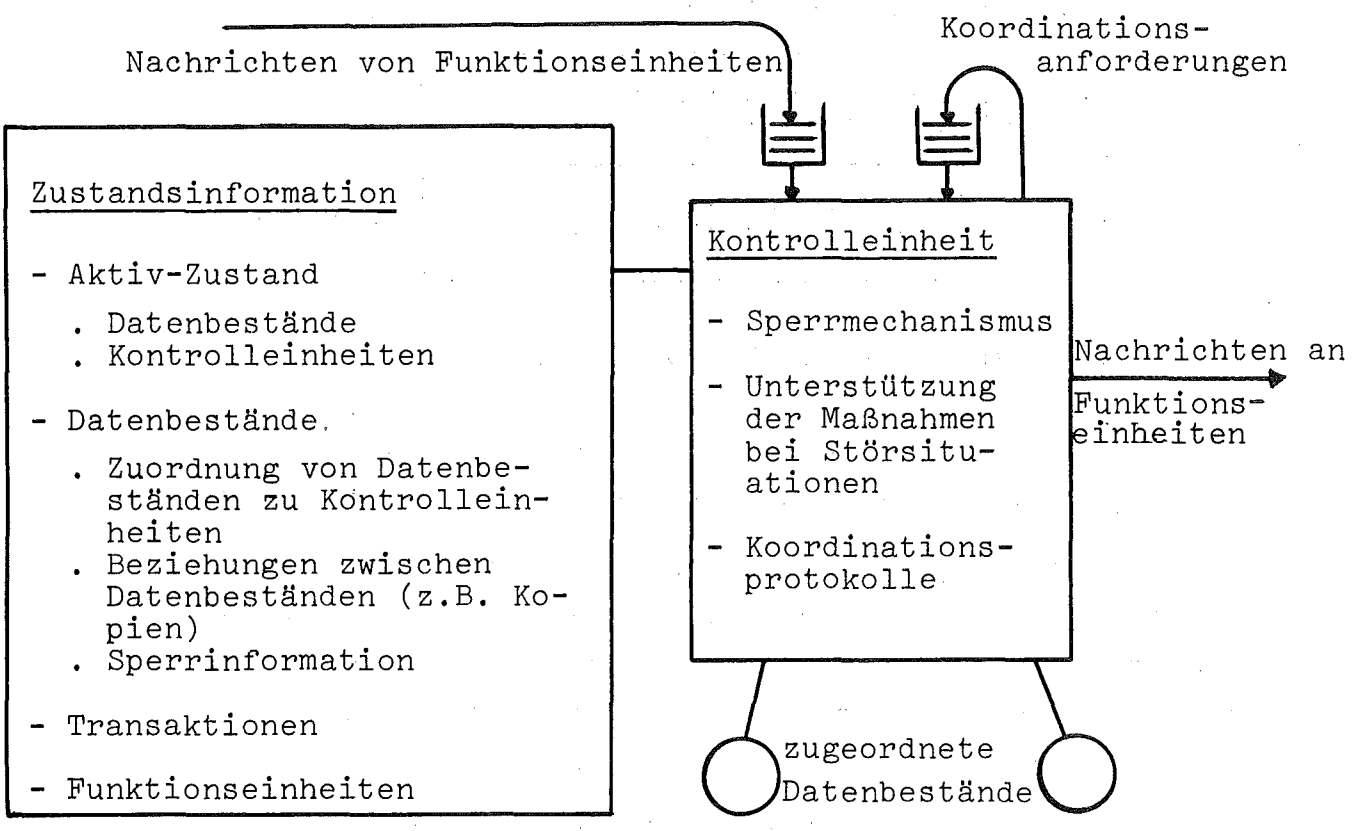


Bild 6.1: Prinzipielle Struktur einer Kontrolleinheit

Die Koordinierung der Aktivitäten der Kontrolleinheiten ist Aufgabe der Koordinationsprotokolle, spezieller Regeln für den Nachrichtenaustausch zwischen den Kontrolleinheiten.

Aus dem breiten Spektrum der Aufgabenbereiche - Sperrung von Datenbeständen, globale Datensicherung, Ausfall und Wiedereingliederung von Datenbeständen und Kontrolleinheiten - lassen sich die drei in Bild 6.2 aufgeführten Aufgabenbereiche als eigenständige Koordinierungsebenen isolieren und hierarchisch anordnen. Die restlichen Aufgabenbereiche können in diese Koordinierungsebenen eingebettet werden.

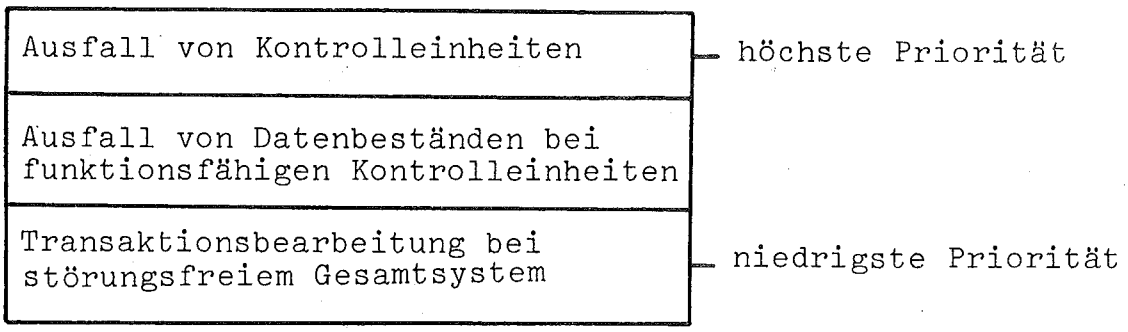


Bild 6.2: Hierarchie der Koordinierungsebenen

Für die Koordinationsprotokolle der verschiedenen Ebenen ist dasselbe Konzept, das sog. Basisprotokoll, anwendbar.

Auf der Grundlage dieser Hierarchie und der Anwendung der Prinzipien des Basisprotokolls in jeder Koordinierungsebene lassen sich für den Übergang zwischen den Koordinierungsebenen die Regeln angeben, die eine verklemmungsfreie Kooperation der Kontrolleinheiten gewährleisten /DRO1/.

Die Anwendung neuer Systemtechnologien führt zu effizienten technischen Lösungen für die Realisierung der Koordinierungsmechanismen.

6.2 Das Basisprotokoll

Das Basisprotokoll beruht auf einer engen Kopplung der Aktionen aller Kontrolleinheiten innerhalb eines Koordinationszyklus. Die Grundzüge seines Ablaufs in einer Kontrolleinheit zeigt Bild 6.3.

Nach Initialisierung eines Koordinationszyklus aus dem Grundzustand (Zustand 1) wartet eine Kontrolleinheit, bei der eine Koordinationsanforderung beantragt wird, in der Abstimmungsphase (Zustand 2) auf die Bereitschaftserklärungen der restlichen Kontrolleinheiten, die Koordinationsanforderung auszuführen, zu deren Durchführung sie sich gegenüber den anderen Kontrolleinheiten ihrerseits bereit erklärt hat. Nach Erhalt aller Bereitschaftserklärungen geht sie in den kritischen Zustand (Zustand 3) über.

Zur Lösung von Konfliktsituationen bei unabhängiger Initialisierung eines Koordinationszyklus durch mehrere Kontrolleinheiten für unterschiedliche Koordinationsanforderungen ist eine Prioritätsregelung vorgesehen. An dieser Prioritätsregelung orientiert sich die Verdrängung von in Abstimmung befindlichen Koordinationsanforderungen niedrigerer Priorität durch diejenige mit der höchsten Priorität; eine Verdrängung ist nur in den Zuständen 1 und 2 erlaubt.

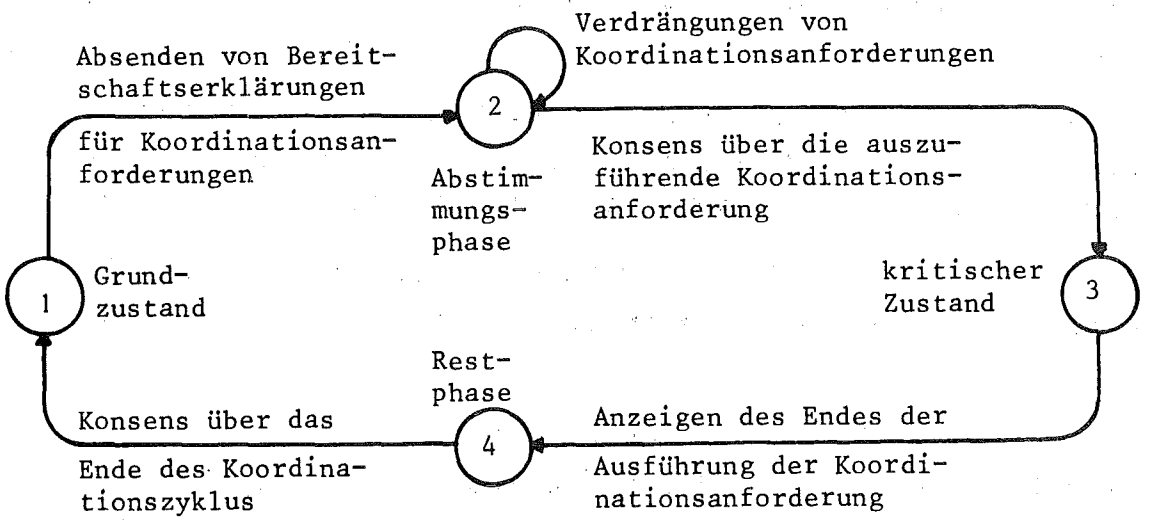


Bild 6.3: Grundzüge des Ablaufs des Basisprotokolls in einer Kontrolleinheit

Im kritischen Zustand wird die allseits akzeptierte Koordinationsanforderung ausgeführt und deren Beendigung den anderen Kontrolleinheiten angezeigt. Über die Restphase (Zustand 4) kehren die Kontrolleinheiten gemeinsam in den Grundzustand zurück.

Das Basisprotokoll gewährleistet /DRO2, HOL1/, daß bei störungsfreiem Gesamtsystem

- alle Kontrolleinheiten in den kritischen Zustand übergehen, falls irgendeine Kontrolleinheit dies tut,
- alle Kontrolleinheiten im kritischen Zustand dieselben Koordinationsanforderungen ausführen,
- ein neuer Koordinationszyklus erst gestartet wird, wenn alle

Kontrolleinheiten den kritischen Zustand verlassen haben.

6.3 Transaktionsbearbeitung bei störungsfreiem Gesamtsystem

Zur Sicherung der operationalen Integrität bei paralleler Ausführung unterschiedlicher Transaktionen und störungsfreiem Gesamtsystem ist - vgl. Bild 6.1 - in jeder Kontrolleinheit vorzugeben:

- ein Sperrmechanismus,
- ein Koordinationsprotokoll zur Koordinierung der Aktivitäten der Sperrmechanismen.

Zur Erläuterung der prinzipiellen Vorgehensweisen bei der Koordinierung der Transaktionsbearbeitung werden folgende Vereinfachungen hinsichtlich der Sperrmechanismen vorausgesetzt:

- Datenbestände entsprechen G'-Dateien; diese seien auch die Einheiten für die Zuweisung an Transaktionen und damit für die Sperrung.
- die Zugriffsverwaltung für logische Dateien gibt zu jeder von einer Transaktion benötigten G'-Datei den Namen und die Sperrungsstufe (Lesesperrung für lesenden und Veränderungssperrung für verändernden Zugriff) an. Die Kontrolleinheiten ergänzen die Anweisungen für Sperrung und Freigabe unter Berücksichtigung der Existenz von Kopien; mit der Sperrung einer G'-Datei sind aus Konsistenzgründen ihre Kopien mit derselben Sperrungsstufe zu belegen.

  Die erarbeiteten Verfahren können jedoch auch zur Prädikatssperrung /ESW2/ verwendet werden, falls z.B. die Disjunktheit der über Prädikate spezifizierten Konsistenzbereiche durch Vergleich der Prädikate ohne Untersuchung des Inhalts der Datenbestände festgestellt werden kann.

Der globale Zustand aller Datenbestände setzt sich zusammen aus:

- der Zuordnung von Datenbeständen zu Kontrolleinheiten,
- der Existenz von Kopien von Datenbeständen,
- der Sperrinformation: aktuelle Sperrungsstufe eines jeden Datenbestandes, Zuordnung der Datenbestände zu Transaktionen.

Der globale Zustand der Transaktionsbearbeitung umfaßt:

- den aktuellen Bearbeitungszustand einer jeden Transaktion; im einfachsten Fall lassen sich für eine Transaktion folgende Zustände angeben:
  . Reservieren von Datenbeständen mit der geforderten Sperrungsstufe,
  . Manipulieren (Lesen, Verändern) von Datenbeständen,
  . Freigeben von Datenbeständen,
- Spezifikationen bzgl. zu sperrender Datenbestände einer jeden Transaktion,
- Wartebeziehungen zwischen Transaktionen bzgl. der Reservierung von Datenbeständen.

Die von den Kontrolleinheiten zu bearbeitenden Koordinationsanforderungen resultieren im wesentlichen aus der Art des Sperrmechanismus einer jeden Kontrolleinheit und den Anforderungen seitens der Transaktionen zur Durchführung von Transaktionszustandsübergängen, in die mehrere Kontrolleinheiten zu involvieren sind. Unterschiedliche Verfahren für die Kooperation der Kontrolleinheiten ergeben sich je nach vorgeplantem Kenntnisumfang einer jeden Kontrolleinheit über den globalen Zustand aller Datenbestände und der Transaktionsbearbeitung.

Es werden zwei Fälle unterschieden:

1. Jede Kontrolleinheit besitzt die vollständige Kenntnis über den globalen Zustand.
2. Jede Kontrolleinheit führt nur eine Teilsicht des globalen Zustands.

In beiden Fällen sind zusätzliche Unterfälle zu berücksichtigen, die sich aus dem jeweils angewandten Sperrmechanismus ergeben /DRO1/.

Fall 1:

Die Zustandsinformation, aufgrund der die Kontrolleinheiten die Transaktionsbearbeitung leiten, liegt bei jeder Kontrolleinheit als Kopie vor. Eine verklemmungsfreie, die opera-

tionale Integrität sichernde Kooperation der Kontrolleinheiten ist daher nur gewährleistet, wenn Veränderungen der Zustandsinformation durch die Kontrolleinheiten nicht zu Konsistenzverletzungen dieses Kopien-Systems führen. Dies erfordert, daß der Sperrmechanismus in allen Kontrolleinheiten gleich ist und die Aktivitäten der Kontrolleinheiten in gegenseitiger Abstimmung erfolgen.

Ein Koordinationsprotokoll ergibt sich durch unmittelbare Anwendung des Basisprotokolls; konkurrierende Koordinationsanforderungen sind u.a.:

- Sperrung von Datenbeständen für eine Transaktion; die Initialisierung eines Koordinationszyklus darf nur erfolgen, falls keiner der benötigten Datenbestände mit einer kollidierenden Sperrungsstufe belegt ist.
- Freigeben von Datenbeständen.

Modifikationen des Koordinationsprotokolls sind notwendig, falls z.B. sukzessive Reservierung von Datenbeständen durch Transaktionen erlaubt ist (Verklemmungsgefahr).

Fall 2:

Die bei einer Kontrolleinheit vorliegende Information über den globalen Zustand umfasse nur

- die Datenbestände und ihre Zuordnung zu den Kontrolleinheiten sowie die Existenz von Kopien,
- die Sperrinformation über die ihr zugehörenden Datenbestände und deren Zuordnung zu in Bearbeitung befindlichen Transaktionen,
- den Bearbeitungszustand aller Transaktionen, in deren Bearbeitung sie involviert ist.

In diesem Fall ist eine mehr entkoppelte Arbeitsweise der Kontrolleinheiten möglich. Die Koordination der Änderungen der Zustandsinformation kann sich auf die unmittelbar betroffenen Kontrolleinheiten beschränken; sogar die Sperrmechanismen der Kontrolleinheiten können differieren.

Da für eine Transaktion u.U. bei mehreren Kontrolleinheiten Datenbestände simultan zu reservieren sind und eine Kontrolleinheit nicht den vollständigen globalen Zustand kennt, können Situationen systemweiter gegenseitiger Blockierungen von Transaktionen entstehen; diese erfordern ein gemeinsames, eng gekoppeltes Handeln von Kontrolleinheiten.

Für ein Koordinationsprotokoll ist daher die in Bild 6.4 dargestellte Grobstruktur ableitbar:

- äußerer Zyklus: Diese Aktionen erfolgen in enger Kopplung aller Kontrolleinheiten (problemorientierte Anwendung des Basisprotokolls),
- innerer Zyklus: Diese Aktionen erfolgen nicht in enger Kopplung mit anderen Kontrolleinheiten.

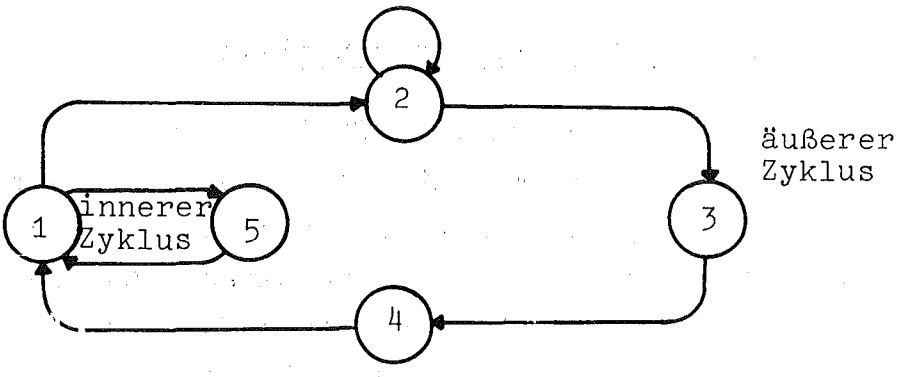


Bild 6.4: Grobstruktur des Koordinationsprotokolls für Fall 2

Unterschiedliche Interpretationen der Aktionen im inneren und äußeren Zyklus ergeben sich aus der gewählten Vorgehensweise zur globalen Regelung von Verklemmungssituationen /DRO1/, z.B.:

- Verhindern von Verklemmungen durch globale Transaktionsvorsortierung,
- Zulassen, Erkennen und Beseitigen von Verklemmungen: Diese wurde für das in 5. vorgestellte operationale Modell ausgewählt.

Die Bedeutung der Verfahren von Fall 1 und 2 liegt darin /DRO1/, daß sie zur Funktionsfähigkeit keine Einschränkungen bezüglich der Struktur und der Verteilung der Daten sowie der Komplexität des Transaktionsaufbaus voraussetzen und folgende Randbedingungen berücksichtigen:

- die koordinierte Ausführung von Transaktionen durch mehrere Rechner,
- die Koordinierung von Teilmengen von Kontrolleinheiten,
- die parallele Bearbeitung unabhängiger, nicht kollidierender Transaktionen,
- die Anforderungen unterschiedlicher Sperrmechanismen.

Für jedes Verfahren lassen sich Einsatzbereiche abgrenzen /DRO1/, in denen sein Koordinierungsaufwand im Vergleich zu dem der anderen Verfahren minimal ist. Weiterführende Untersuchungen des Leistungsverhaltens der Verfahren sind hier jedoch noch erforderlich.

6.4 Fehlertolerante Koordinierungsmechanismen

Störsituationen können die Funktionsfähigkeit und Leistungsfähigkeit des Gesamtsystems wesentlich mindern, falls es nicht fehlertolerant ausgelegt ist. Für verteilte Daten bedeutet dies, daß eine redundante Auslegung von Datenbeständen und Kontrolleinheiten notwendige Voraussetzung für eine hohe Verfügbarkeit des Gesamtsystems ist. In Ergänzung hierzu sind die unter der Annahme eines zuverlässigen Gesamtsystems entwickelten Koordinationsprotokolle zu "fehlertoleranten" Protokollen auszubauen, die die bei Ausfall und Wiedereingliederung von Datenbeständen und Kontrolleinheiten vom System zu ergreifenden Maßnahmen unterstützen.

Solche Maßnahmen haben zum Ziel:

- Erkennung des Ausfalls von Datenbeständen: Dies ist Aufgabe des lokalen Betriebssystems. Die Ausfallmeldung wird einer Kontrolleinheit über die Zugriffsverwaltung für G'-Dateien zugeleitet.
- Erkennung des Ausfalls von Kontrolleinheiten: Die Kontrolleinheiten führen eine gegenseitige Zeitüberwachung ("time-outs")

hinsichtlich protokollbedingter Reaktionen durch.

- Sicherung der operationalen Integrität für alle Transaktionen und Datenbestände des Restsystems über den Störfall hinweg: Zur globalen Datensicherung müssen netzweit gültige Wiederaufsetzpunkte erstellt und Transaktionsjournale geführt werden. Die Erstellung eines solchen Wiederaufsetzpunkts ist eine Koordinationsanforderung, die in die Koordinierungsebene für Transaktionsbearbeitung bei störungsfreiem Gesamtsystem einzubetten und in Konkurrenz zu den übrigen Koordinationsanforderungen zu bearbeiten ist.
- Rekonfiguration aller arbeitsfähigen Komponenten zu einem funktionsfähigen Restsystem: Diese ist unmittelbar mit der Koordinierung über die Änderung des Aktiv-Zustands über Datenbestände und Kontrolleinheiten sowie evtl. weiterer Zustandsinformation in der entsprechenden Koordinierungsebene verbunden.
- Sicherung der inneren Beziehungen von Daten für die Wiedereingliederung von Datenbeständen: Vorgehensweisen beinhalten die Sperrung aller mit dem ausgefallenen Datenbestand in inneren Beziehungen stehenden Datenbeständen gegenüber Veränderungen sowie das Übersenden eines aktuellen Datenbestandes bei Wiedereingliederung, falls eine Wiederherstellung nur über Kopien auf anderen Rechnern möglich ist. Die Wiedereingliederung ist eine Koordinationsanforderung, die ebenfalls in der Koordinierungsebene für Transaktionsbearbeitung bei störungsfreiem Gesamtsystem anzusiedeln ist.

Das Basisprotokoll läßt sich in der Koordinierungsebene zur Reaktion auf Ausfall von Datenbeständen bei funktionsfähig verbleibenden Kontrolleinheiten unmittelbar einsetzen. Mit der koordinierten Änderung des Aktiv-Zustands über Datenbestände sind zusätzliche Maßnahmen zu verbinden wie z.B. Rücksetzen von Transaktionen, deren weitere Bearbeitung wegen des Ausfalls nicht mehr möglich ist.

Zur Koordinierung der Änderung des Aktiv-Zustands über Kontrolleinheiten bei Ausfall einer Kontrolleinheit ist das

Basisprotokoll zu modifizieren. Während eines Koordinationszyklus zur Ausfallbehandlung können zusätzlich Kontrolleinheiten ausfallen. Der Übergang vom Zustand 2 in 3 wird auch ausgeführt, falls nach Ablauf der Zeitschranke("time-out") noch kein Konsens vorliegt. Im Zustand 3 wird ein vorläufig als neu zu betrachtender Aktiv-Zustand ermittelt und den restlichen funktionsfähigen Kontrolleinheiten mitgeteilt. Bei Übereinstimmung aller noch funktionsfähigen Kontrolleinheiten wird der ermittelte neue Aktiv-Zustand übernommen; ansonsten wird ein neuer Koordinationszyklus in dieser Koordinierungsebene initiiert. Das Verfahren endet entweder mit der erfolgreichen Bildung eines funktionsfähigen, die Weiterführung der Transaktionsbearbeitung sicherstellenden Teilsystems von Kontrolleinheiten, oder mit dem Übergang aller Kontrolleinheiten in den Ausfallzustand, der ein manuelles Eingreifen erfordert /DRO1/.

Die Wiedereingliederung einer Kontrolleinheit wird von den aktiven Kontrolleinheiten in der Koordinierungsebene für Transaktionsbearbeitung bei störungsfreiem Gesamtsystem bearbeitet; die wiedereinzugliedernde Kontrolleinheit bleibt währenddessen in der für die Ausfallbehandlung von Kontrolleinheiten zuständigen Koordinierungsebene, die hierzu entsprechend zu erweitern ist (siehe /DRO1/).

6.5 Realisierung von Koordinierungsmechanismen

Die naheliegende Realisierung der vorgestellten Koordinierungsmechanismen besteht in der vollständigen Integration der Kontrolleinheiten in die Arbeitsrechner eines verteilten DV-Systems.

Eine wesentliche Grundlage hierfür ist der Grad an Komfort der Schnittstelle, die das Nachrichtentransportsystem den Kontrolleinheiten zum Nachrichtenaustausch bereitstellt. Von den existierenden Alternativen von Nachrichtentransportsystemen eignet sich für die Kooperation von Kontrolleinheiten insbesondere das Konzept der verbindungsorientierten Kommunikation (siehe Kap. 7).

Die vollständige Integration von Kontrolleinheiten in die Arbeitsrechner bedeutet eine nicht unbeträchtliche Belastung der Arbeitsrechner. Es empfiehlt sich, Teile der Koordinierungsmechanismen in speziellen Komponenten des Nachrichtentransportsystems zu realisieren; insbesondere bietet sich eine Integration in Kommunikationsprozessoren an.

Dies kann in der Form spezieller Koordinations-Tasks, Sekretäre, geschehen, die eine ähnliche Struktur wie die Kontrolleinheiten aufweisen und für die Kooperation äquivalente Protokolle benutzen. Funktionen, die den Sekretären übertragen werden können, sind u.a.

- Protokollfunktionen,
- Sperrmechanismus: Führen von Zustandsinformation,
- "time-out"-Kontrolle.

Zur Effizienzuntersuchung wurden für das Basisprotokoll beide Realisierungsvarianten als Simulationsmodelle implementiert. Um validierbare Aussagen zu gewinnen, wurde ein realitätsnahes Modell des Nachrichtentransportsystems eines existierenden Rechnernetzes verwendet /STS1/+). Die Experimente zeigten eine generelle Überlegenheit der Sekretärstechnik sowohl hinsichtlich der Belastung des verteilten DV-Systems durch das Basisprotokoll als auch der Festlegung optimaler Zeitschranken für die Überwachung der Aktionen der Kontrolleinheiten, um deren verklemmungsfreie Kooperation zu gewährleisten /DRO1/.

+) Das Modellierungssystem für das Nachrichtentransportsystem wurde im Auftrag des Hahn-Meitner-Instituts (Berlin) am IDT entwickelt.

7. Ein Nachrichtentransportsystem für die Interkommunikation zwischen den funktionellen Komponenten verteilter Datenbanken

7.1 Anforderungen

Die anhand des operationalen Modells für die verteilte Dateiverwaltung identifizierten funktionellen Komponenten werden durch ein System verteilter Prozesse realisiert. Die Form der Interkommunikation dieser Prozesse prägt sowohl die erforderliche Konzipierung der prozeßseitigen Schnittstelle des Nachrichtentransportsystems als auch den grundsätzlichen Ansatz zur Interprozeßkommunikation.

Die während der Betriebsphasen eines verteilten DV-Systems stabile Konfiguration der für die Datenverwaltung zuständigen Prozesse läßt den verbindungsorientierten Ansatz zur Interprozeßkommunikation geeigneter erscheinen als den nachrichtenorientierten Ansatz /ABS1/. Die verbindungsorientierte Interprozeßkommunikation sieht die Einrichtung logischer Kanäle zwischen Prozessen vor; die Lebensdauer dieser Kanäle ist von der Dauer und Intensität des Nachrichtenaustauschs unabhängig.

Die funktionellen Anforderungen an die Gestaltung der Schnittstelle für Interprozeßkommunikation sind im einzelnen:

- Grundfunktionen
  Neben Funktionen für den elementaren Nachrichtentransport (Senden und Empfangen) müssen Kommunikationsprimitive zum Auf- und Abbau logischer Kanäle bereitgestellt werden.
- Erweiterte Übertragungsfunktionen
  Die volle Nutzung der Möglichkeiten der Parallelverarbeitung erfordert Funktionen zur Initialisierung der parallelen Übertragung identischer Nachrichten an mehrere Prozesse (broadcasting). Dies führt insbesondere bei der Durchführung von Koordinationsaufgaben zur Effizienzverbesserung.
- Ereignissynchronisation
  Rückmeldungen über Abschluß und Verlauf der Bearbeitung von Kommunikationsanforderungen sollten mit einer effizienten Ereignissynchronisation verbunden sein, die den wartenden

Prozeß von ständigen Zustandsabfragen ("busy wait") befreit und wahlweise die Suspendierung oder das zur Abwicklung des Kommunikationsvorganges parallele Verbleiben des Prozesses im aktiven Zustand ermöglicht.

- Asynchrone Verarbeitung
  Auftragsbeziehungen zwischen Prozessen erfordern die Möglichkeit des asynchronen Nachrichtenempfangs (bezogen auf den Prozeßablauf) und der Entkopplung von Sende- und Empfangsvorgängen durch Verwendung gerichteter logischer Kanäle, mit deren Hilfe z.B. Vollduplexverkehr realisiert werden kann.
- Variable Nachrichtenlängen
  Das operationale Modell für die verteilte Dateiverwaltung unterscheidet beim Nachrichtenaustausch zwischen Prozessen die Übertragung von Anweisungen, Kontrollnachrichten und Datenblöcken. Entsprechend resultieren stark variierende Nachrichtenlängen, denen von Seiten des Nachrichtentransportsystems durch eine geeignete Pufferverwaltung begegnet werden muß.
- Prioritätsgesteuerte Übertragung
  Ein reaktionsschnelles Systemverhalten wird erreicht, wenn der besonderen Übertragungspriorität von bestimmten Kontrollnachrichten und speziellen Anweisungen, beides in der Regel Kurznachrichten, Rechnung getragen werden kann.
- Identifikation
  Die eindeutige Identifikation der kommunizierenden Prozesse bzw. ihrer Ein- und Ausgänge (Ports) für logische Kanäle muß gewährleistet sein, bei weitgehend automatischer Ortsbestimmung der Empfangsprozesse.

Die Realisierung der Interprozeßkommunikation sollte in zweifacher Hinsicht einheitlich erfolgen:

- die Kommunikation zwischen Prozessen im selben Rechner ("intra host communication") sollte über die gleiche Schnittstelle möglich sein wie die Kommunikation zwischen Prozessen in verschiedenen Rechnern ("inter host communication").
- bei heterogenem Aufbau des verteilten DV-Systems ist eine

systeminterne Standardschnittstelle für Interprozeßkommunikation erforderlich, wenn die übertragbare Realisierung und freie Rekonfigurierbarkeit der Funktionseinheiten des Dateiverwaltungssystems angestrebt wird.

Beide Anforderungen gemeinsam implizieren eine einheitliche, von der speziellen Architektur des verteilten Systems unabhängige Schnittstelle für Interprozeßkommunikation.

Eine weitere Randbedingung, die das Konzept eines Nachrichtentransportsystems wesentlich beeinflußt, ist die Einbeziehung von Kleinrechnern als Arbeitsrechner. Die begrenzte Leistungsfähigkeit dieser Rechnerkategorie verlangt, daß die Arbeitsrechner weitgehend von dem durch die Kommunikation bedingten Verwaltungsaufwand entlastet werden. Dies gelingt nur durch Auslagerung von Teilen des Nachrichtentransportsystems auf spezielle Hardware, auf sog. Kommunikationsprozessoren.

7.2 Gewähltes Konzept

Die Grobstruktur eines Nachrichtentransportsystems, das die in 7.1 gestellten Forderungen erfüllt, zeigt Bild 7.1. Die Grundkonzeption des Systems, das eine systemweit einheitliche, von der physikalischen Realisierung des Nachrichtentransports unabhängige Schnittstelle für die Interprozeßkommunikation (IPC) zur Verfügung stellt, resultiert aus der konsequenten Verwertung der in Großrechnernetzen gesammelten Erfahrungen /CHM1, CHU2/.

Als wesentliche Komponenten des Nachrichtentransportsystems können die Transportstation und das physikalische Transportsystem (oft als Kommunikationssubsystem bezeichnet) identifiziert werden.

Der Transportstation obliegt die Aufgabe der Anpassung der IPC-Schnittstelle an möglicherweise nicht einheitliche Schnittstellen des physikalischen Transportsystems; dabei bedient sie sich der an der Betriebssystemschnittstelle verfügbaren Betriebssystemfunktionen (Treiberaufrufe, Synchronisationsfunktionen, etc.). Die Transportstation wird in den Arbeitsrechnern des verteilten DV-Systems als Systemprozess reali-

siert, der die Kommunikationsanforderungen von Benutzer- und anderen Systemprozessen entgegennimmt.

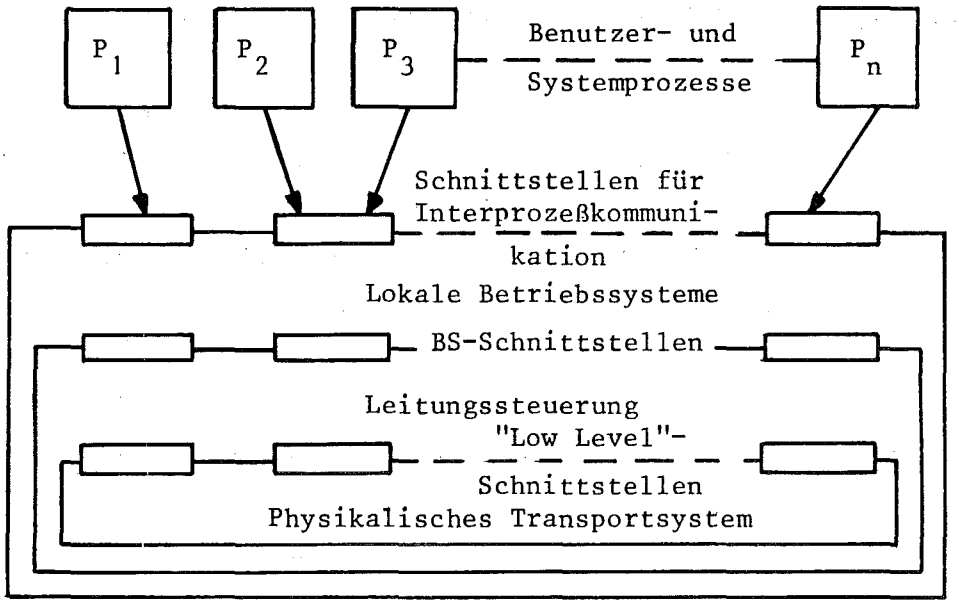


Bild 7.1: Grobstruktur eines Nachrichtentransportsystems

Als elementare Grundaufgabe übernimmt die Transportstation den Transfer beliebig langer Nachrichten zwischen Prozessen über vor Beginn eines Dialogs eingerichtete logische Kanäle. Sie fungiert dabei als Multiplexer/Demultiplexer, der die eindeutige Zuordnung der ein- und ausfließenden Nachrichtenströme zu den angegebenen Sende- und Zielprozessen herstellt und sicherstellt.

Soweit notwendig werden zu übertragende Nachrichtentexte von der sendenden Transportstation in übertragungsgerechte Pakete fragmentiert. Diese werden, mit zusätzlicher Kontrollinformation versehen, dem autonom arbeitenden Kommunikationssubsystem zum Transport übergeben. Umgekehrt stellt eine Transportstation als Empfänger einfließende Pakete wieder zu Nachrichtentexten zusammen und übergibt sie dem vorgesehenen Zielprozeß.

Transportstationen in den unterschiedlichen Arbeitsrechnern

kooperieren auf der Basis der im verteilten DV-System einheitlich vereinbarten Regeln des Transportprotokolls. Die formalen Regeln des Transportprotokolls werden durch einen in die Transportstation integrierten endlichen Automaten, eine sog. Protokolleinheit /DAB1/, realisiert.

Ein Flußkontrollmechanismus in jeder Transportstation reguliert den in einen Arbeitsrechner eingehenden Nachrichtenstrom. Gerade im Kleinrechnerbereich mit meist beschränkten Adreßräumen und Pufferspeichern ist ein solcher Kontrollmechanismus für einen koordinierten Ablauf von größter Bedeutung. Flußkontrolle geschieht deshalb explizit durch eine dem eigentlichen Transfer vorangehende Abstimmung zwischen Empfänger und Sender. Erst eine Sendeaufforderung seitens des Empfängers, der einen Puffer für die zu erwartenden Daten reserviert hat, löst eine Nachrichtenübertragung zwischen Partnerprozessen aus.

Da in einem verteilten DV-System mit heterogenen Arbeitsrechnern jedes Betriebssystem eine ihm eigene und zumeist zu anderen Betriebssystemen inkompatible Namensgebung für seine Prozesse, Betriebsmittel und E/A-Aktivitäten hat, muß ein neuer, netzweit eindeutiger Namensraum geschaffen werden. Die Elemente dieses neuen Namensraums sind identisch mit den in 2.5 eingeführten Ports.

Vom Kommunikationsstandpunkt aus betrachtet ist eine Transportstation zunächst nichts anderes als eine Gruppierung von Ports, die durch je einen Bitstring definierter Länge identifiziert werden. In einer Transportstation werden die Portnamen mit den Elementen des lokal gültigen Namensraums assoziiert. Die Abbildung wird dabei dynamisch nach den Bedürfnissen der Prozesse, die die Dienstleistung der Transportstation in Anspruch nehmen wollen, vorgenommen.

Der Bitstring, durch den ein Prozeß über einen Portnamen identifiziert werden soll, muß so gewählt werden, daß sich alle logischen Namen eindeutig auf den Portadreßraum abbilden lassen. Die niederwertigen Bits werden dabei zur Unterscheidung einzelner Prozesse, die höherwertigen Bits zur Identifikation verschiedener Arbeitsrechner herangezogen.

Untermengen dieses hierarchischen Adreßraums können für Dienstleistungsprozesse, die z.B. andere Benutzerprozesse initialisieren, reserviert werden. Ports, die miteinander kommunizieren wollen, werden miteinander verknüpft. Diese Assoziation eines Portpaares wird durch die Eröffnungsphase des Transportprotokolls vorgenommen und überwacht. Kommt die Verknüpfung zweier Ports zustande, können die assoziierten Prozesse unter Beziehung auf diese Portnamen über den neugeschaffenen logischen Kanal Nachrichten austauschen. Dieser neue logische Kanal wird eindeutig durch das Tupel beider Portadressen identifiziert (Bild 7.2).

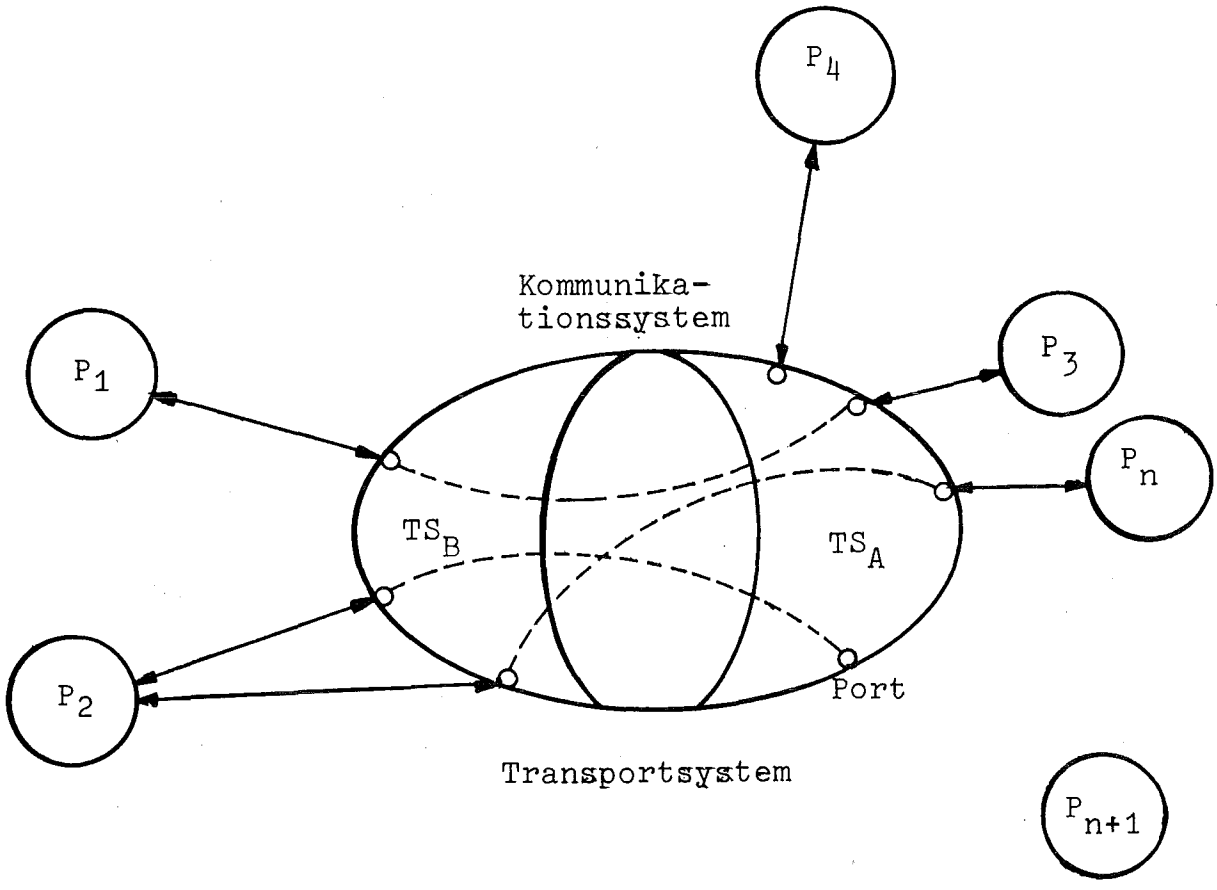


Bild 7.2: Verbindungsorientierte Kommunikation

Das physikalische Transportsystem umfaßt alle Vorrichtungen, die für die fehlerfreie Übertragung von Daten zwischen räumlich getrennten Arbeitsrechnern notwendig sind. Dazu gehören Einrichtungen, die die zu übertragenden Nachrichten in

einer der Übertragungstechnik angepaßten Form codieren bzw. decodieren und für die Abwicklung der Leitungsprozeduren zuständig sind.

Leitungsprozeduren definieren die beim physikalischen Datentransport geltenden Regeln, nach denen der Verbindungsauf- und -abbau, die Steuerung des Nachrichtenflusses, Fehlererkennung, Übertragungswiederholung und Wiederanlauf erfolgen.

Für die Realisierung des physikalischen Transportsystems wurde ein Ansatz gewählt, der eine weitgehende Entlastung der Arbeitsrechner von elementaren Kommunikationsaufgaben erlaubt und somit insbesondere die Einbeziehung leistungsschwacher Kleinrechner in verteilte DV-Systeme ermöglicht:

Auf der Grundlage von Mikroprozessoren lassen sich sog. Kommunikationsprozessoren aufbauen, denen die gesamte Abwicklung der Leitungsprozeduren und Vermittlungsaufgaben übertragen werden kann. Im Gegensatz zu anderen Lösungen (vgl. /TOT1/) lassen sich hier bei vertretbarem Kostenaufwand (1000 bis 3000 DM pro Kommunikationsprozessor) maßgeschneiderte Lösungen verwirklichen, die durch die freie Programmierbarkeit der Mikroprozessoren leicht an unterschiedliche Schnittstellen der Arbeitsrechner, Konfigurations- und Leitungsprotokolländerungen, und an Änderungen der Routing-Strategie für die Vermittlung von "Paketen" angepaßt werden können.

Bei geringen Anforderungen an die Übertragungsleistung ist es durchaus möglich, selbst elementare Aufgaben wie etwa die Serialisierung von Nachrichten vom Mikroprozessor ausführen zu lassen.

Bei hohem Nachrichtendurchsatz, wie er z.B. in lokalen verteilten DV-Systemen verlangt wird, ergeben sich hierbei jedoch Zeitprobleme, die eine Unterstützung des Mikroprozessors durch vorgeschaltete Hardwareeinrichtungen für Serialisierung, Codierung, Blockpufferung und Prüfcodegenerierung erfordern /PAT1/.

Bild 7.3 zeigt die Grobstruktur eines Kommunikationsprozessors mit dieser Konzeption /KRK1/. Er besteht aus zwei

unterschiedlichen Komponenten:

- dem Prozessorteil mit Zentraleinheit, Programm- und Datenspeicher, Ein-/Ausgabe,
- dem oder den Leitungsinterface(s), die die physikalische Ankopplung an die Übertragungsleitung(en) leisten.

Im einzelnen sind die Aufgaben dabei folgendermaßen verteilt /HKK1, KRK1/:

Prozessorteil:

- Kommunikation mit dem angeschlossenen Arbeitsrechner über eine Parallelschnittstelle,
- Abwicklung der Leitungsprozedur auf Blockebene (Generierung der Kontrollnachrichten, Reaktion auf Übertragungsfehler, Zeitüberwachung),
- Steuerung des oder der angeschlossenen Leitungsinterfaces ("Sende Block", "Gehe in Grundzustand"), und Verarbeitung derer Zustandsmeldungen ("Block empfangen", "Ü-Fehler aufgetreten", "Block senden fertig" etc.).

Leitungsinterface:

- Puffern von Datenblöcken bei Empfang und Senden,
- Serialisierung, Parallelisierung,
- Prüfcodegenerierung bzw. -test,
- Bit- und Bytesynchronisation,
- Modulation/Demodulation,
- darüber hinaus je nach Prozedur Bit- oder Bytestuffing.

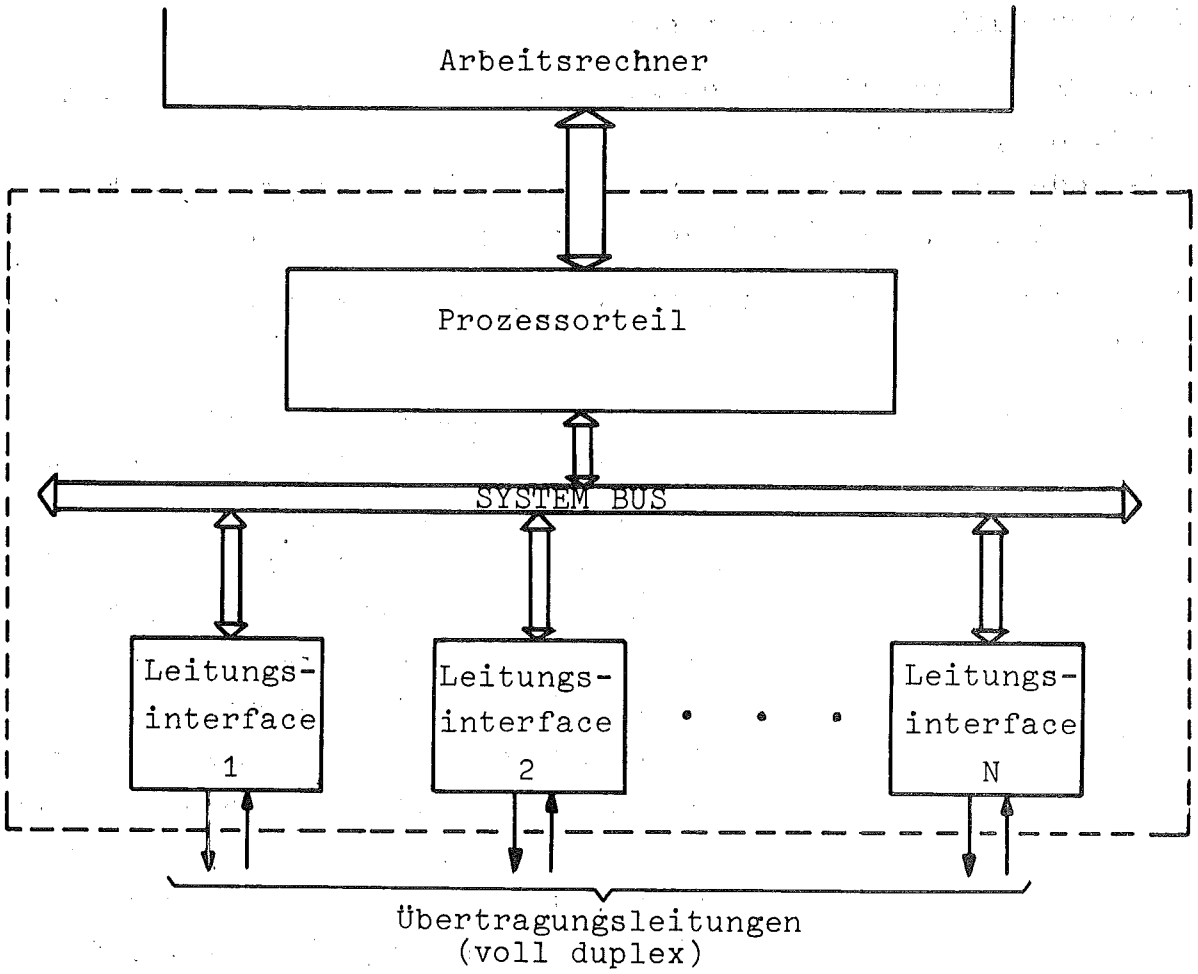


Bild 7.3: Grobstruktur eines Kommunikationsprozessors auf Mikroprozessorbasis

7.3 Pilotimplementierung

Experimentelle Grundlage für die Erprobung funktioneller Komponenten verteilter Datenverwaltungssysteme bildet die Pilotimplementierung eines verteilten DV-Systems am Institut für Datenverarbeitung in der Technik. Bei dem verteilten DV-System handelt es sich um ein lokales System, das in der ersten Ausbauphase zwei Kleinrechner vom Typ PDP11/03 als Arbeitsrechner umfaßt. Als Hintergrundspeicher steht den Arbeitsrechnern jeweils ein Floppy-Disk-Doppellaufwerk zur Verfügung. Für die in der Zukunft geplante Pilotimplementierung einer verteilten Datenbank wird das verteilte DV-System in der zweiten Ausbauphase um zwei Kleinrechnersysteme Siemens PR330 mit Plattenspeichern erweitert werden.

Das verteilte DV-System basiert auf einem Nachrichtentransportsystem, dem das in 7.2 beschriebene Konzept zugrunde liegt. Das physikalische Transportsystem, von der Grundstruktur her ringförmig ausgelegt, ist so realisiert, daß es die Einbeziehung einer nach oben nur durch die verfügbare Übertragungsleistung beschränkte Zahl von Arbeitsrechnern ermöglicht.

Bei der Implementierung der Transportstation wurde eine Abbildung der zu erbringenden Funktionen auf insgesamt 5 selbständige, als Tasks organisierte Module vorgenommen. Bild 7.4 zeigt die sich ergebende Struktur:

- Das Benutzerinterface nimmt die Anforderungen der Benutzerprozesse entgegen, analysiert sie syntaktisch und initialisiert die Beschreibungsblöcke.
- Eine Sendetask bereitet Teilnachrichten ("Pakete") als physikalische Transportelemente auf und übergibt sie dem Kommunikationssubsystem.
- Umgekehrt sammelt eine Empfängertask einlaufende Teilnachrichten, interpretiert die mitgelieferte Kontrollinformation und schleust die Daten in die Puffer der Empfängerprozesse.

Sende- und Empfängertask bedienen sich der Dienste der vom Betriebssystem zur Verfügung gestellten Ein-/Ausgabetreiber.

- Eine unabhängig laufende Uhr überwacht festgesetzte Timeoutschranken und initiiert bei Zeitüberschreitungen die daraus resultierende Fehlerbehandlung.
- Allen vorgenannten Modulen ist ein Taskumschalter übergeordnet, der für die prioritätsgerechte zeitliche Sequenzierung der einzelnen Module verantwortlich ist.

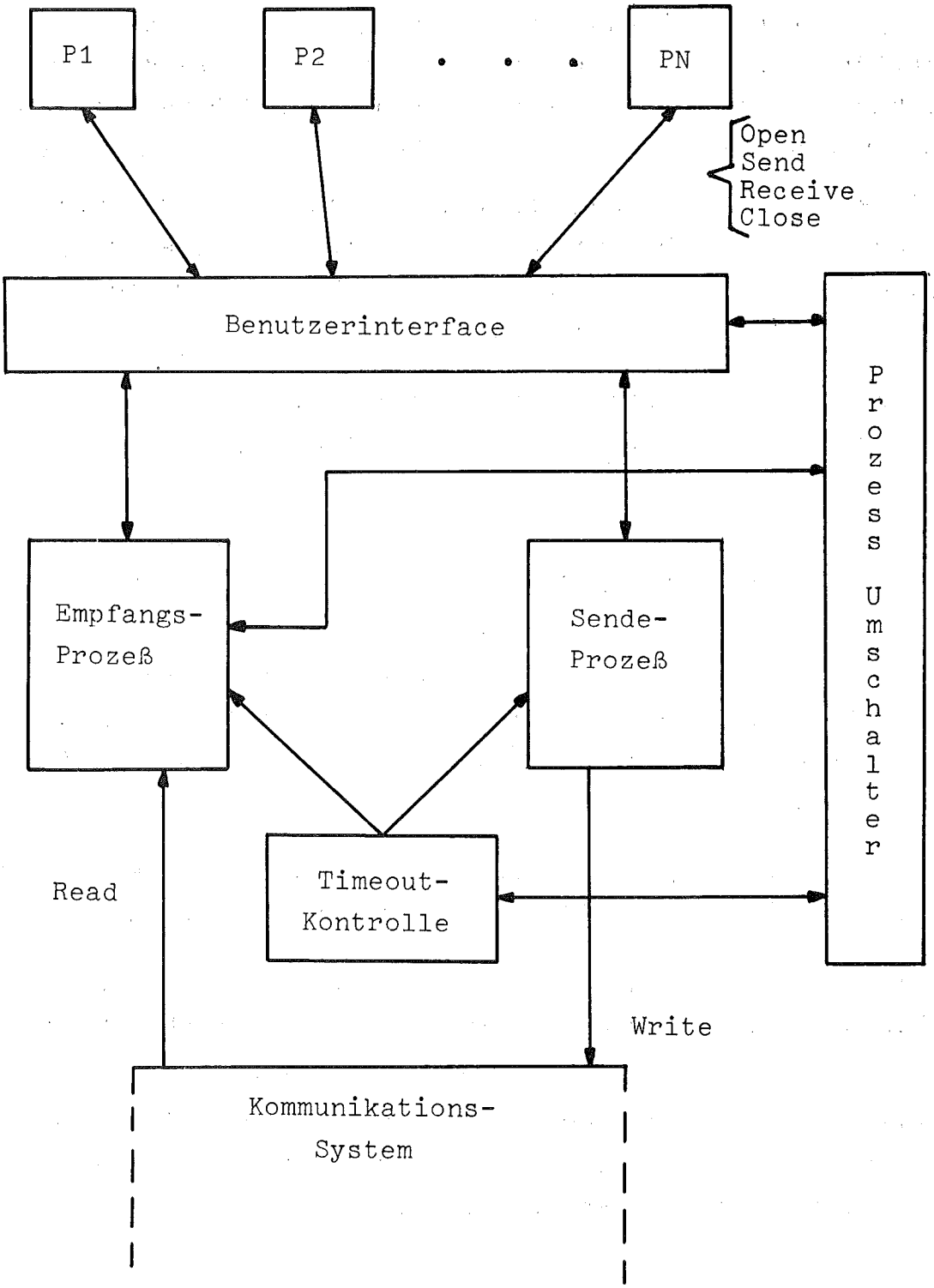


Bild 7.4: Struktur einer Transportstation

Die Transportstation wurde in den als Arbeitsrechnern verwendeten PDP11/03 Rechnern unter Verwendung der Foreground/Background-Version des Betriebssystems RT11 implementiert. Dieses Betriebssystem erlaubt die prioritätsgesteuerte Abarbeitung zweier Prozesse, wobei die Transportstation im Vordergrund die zeitkritischen Anforderungen des physikalischen Transportsystems übernimmt, während der Hintergrund-Laufbereich für Anwendungsprogramme reserviert bleibt. RT11 stellt einen Satz von Betriebssystemfunktionen für Synchronisation und lokale Prozeßkommunikation zur Verfügung, die sich für die Implementierung der Transportstation als hilfreich erweisen.

Die von der Transportstation bereitgestellte Schnittstelle für Interprozeßkommunikation bietet ein Spektrum von Funktionen an, das den in 7.1 zusammengestellten Anforderungen weitgehend entspricht, wenn auch, durch die Implementierungsumgebung bedingt, eine Reihe von Einschränkungen erforderlich waren (paralleles Senden, asynchrone Verarbeitung). Die realisierten Kommunikationsprimitive sind:

- OPEN <port 1>, <port 2>, <t>
  der aufrufende Prozeß fordert von der Transportstation den Aufbau eines logischen Kanals zwischen dem lokalen Port <port 1> und dem entfernten Port <port 2>. <port 1> besteht aus der Konkatenation von lokaler Rechnernummer, die die Transportstation kennt, und der vom Aufrufer angegebenen Portnummer. <port 2> muß vollständig spezifiziert werden. Der Verbindungsaufbau soll innerhalb eines Zeitraums <t> abgeschlossen werden können.

- OPENPORT <port>, <t>
  der aufrufende Prozeß stellt für einen Zeitraum <t> einen Portnamen zur Verfügung, der von Prozessen auf anderen Arbeitsrechnern angewählt werden kann. Diese Funktion erlaubt den Aufbau eines Kommunikationskanals, bei dem seitens des Aufrufers die Identität des Partners noch unbekannt ist. Serviceprozesse und Loggerprozesse /CER1/ werden sich vorzugsweise dieser Dienstleistung bedienen.

- CLOSE <port>
  dieser Aufruf verlangt die Auflösung einer logischen Verbindung, sowie die Freigabe der damit verbundenen Betriebsmittel.

- CLOSEPORT <port>
  deaktiviert ein Adreßelement und gibt die zugeordneten Beschreibungsblöcke frei.

- SENDW <port 1>, <port 2>, <buf>, <l>, <t>

- SEND <port 1>, <port 2>, <buf>, <l>, <t>
  der diesen Aufruf absetzende Prozeß wünscht die Übertragung des an der Adresse <buf> abgelegten Textes der Länge l über die durch das Tupel (<port 1>, <port 2>) spezifizierte Verbindung. Bei SENDW wird der Aufrufer bis zum Abschluß des Transfers suspendiert; bei SEND-Aufrufen bleibt ihm die Kontrolle über den Prozessor, auch nach der Übergabe des Aufrufs an die Transportstation, solange er nicht von höherprioren Prozessen verdrängt wird. Der Transfer des Pufferinhalts soll innerhalb der Zeitschranke <t> erfolgt sein.

- RECEIVEW <port 1>, <port 2>, <buf>, <l>, <t>

- RECEIVE <port 1>, <port 2>, <buf>, <l>, <t>
  der Aufrufer wünscht, daß die nächste zu erwartende Nachricht seitens des Partnerprozesses auf dem angegebenen Kanal an der mit <buf> bezeichneten Adresse abzulegen ist. Der Aufruf kann wiederum synchron (RECEIVEW) oder asynchron (RECEIVE) beantwortet werden.

- TESTSTATUS <port>, <adr 0>, <adr 1>
  diese Funktion dient der Statusabfrage bei asynchronen Kommunikationsprimitiven. Ist einer der mit den Aufrufen SEND oder RECEIVE verbundenen Aufträge von der Transportstation abgeschlossen worden, so verzweigt das rufende Programm zur Marke <adr 1>, andernfalls zur Marke <adr 0>. Ein mitgelieferter Returncode gibt nähere Auskunft über eine evtl. eingetretene Fehlersituation.

Die Angabe einer Zeitschranke bei den Aufrufen ist optional. Beim Fehlen einer Zeitangabe wird eine unbeschränkte Dauer

angenommen.

Der implementierte Satz von Kommunikationsfunktionen erfüllt die Forderung nach Unabhängigkeit von der physikalischen Realisierung des Informationstransports.

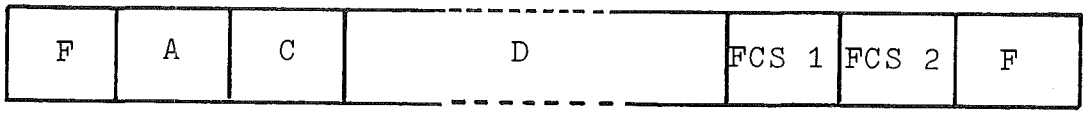
Das physikalische Transportsystem der Pilotimplementierung benutzt auf dem Mikroprozessor INTEL 8080A /INT2/ bzw. ZILOG Z80 /SHI1/ basierende Kommunikationsprozessoren als "Interfaces" zu einem als Bus verwalteten Ringsystem. Der Aufbau der Kommunikationsprozessoren entspricht dem in 7.2 vorgestellten Konzept.

Der Prozessorteil kann bis zu je 4 K Byte ROM und RAM zur Speicherung von Programmen bzw. Daten ausgebaut werden. Das Leitungsinterface, über den Systembus mit dem Mikroprozessor verbunden, ist auf Übertragungsraten von 156 K Bit/sec bis 2,5 M Bit/sec einstellbar und läßt beim Senden und Empfangen die Zwischenpufferung von Datenblöcken bis 128 Byte Nutzdatenlänge zu. Eine detaillierte Beschreibung der Kommunikationsprozessorhardware findet sich in /KRK1/ und /HKK1/.

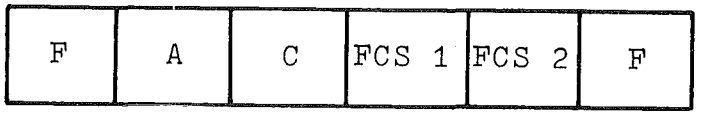
Die Kommunikationsprozessoren sind durch Zweidrahtleitungen miteinander verbunden. Das verwendete Leitungsprotokoll entspricht dem ISO-Vorschlag für die asymmetrische Variante der Leitungsprozedur HDLC /INT1/.

Die verwendete Prozedurvariante bedingt eine hierarchische Organisation des physikalischen Transportsystems, die sich in einer unterschiedlichen Kommunikationssoftware der von der Hardware her identischen Kommunikationsprozessoren niederschlägt, je nach Einsatz des Kommunikationsprozessors als "Haupt-" oder "Unterstation". Im Rahmen des Gesamtkonzepts hat dieser Umstand jedoch keinerlei Auswirkungen auf die Kommunikation der Arbeitsrechner untereinander; das Transportprotokoll für die Kommunikation zwischen den Transportstationen bleibt völlig symmetrisch. Ein für später vorgesehener Austausch der asymmetrischen HDLC-Variante gegen ein dezentralisiertes Leitungsprotokoll bleibt ohne Einfluß auf die "logische" Transportebene, d.h. auf die Ebene der Kommunikationshierarchie oberhalb des physikalischen Transportsystems.

HDLC ist eine bitorientierte Prozedur, die es erlaubt, code-transparente Daten mit Hilfe synchroner Übertragungsverfahren bitseriell zu übertragen. Dies geschieht in Nachrichtenrahmen ("frames") variabler Länge; jeder Rahmen besteht aus mehreren Feldern, deren Bedeutung durch ihre relative Lage zu Anfang bzw. Ende des Rahmens gegeben ist (Bild 7.5).



Informationsformat



Kontrollformat

F = Flagbyte
A = Adressfeld
(Adresse der Unterstation)
C = Kontrollfeld
D = Datenfeld
(Beliebig lang)

FCS = Frame Check Sequence
(CRC-Prüfzeichen)

Bild 7.5: HDLC-Formate ("frames")

Im einzelnen sind die Felder folgendermaßen definiert:

- F Flagbytes begrenzen beidseitig jeden "frame"; sie bestehen aus der Bitfolge 0111 1110. Um ihr zufälliges Auftreten innerhalb eines Rahmens zu verhindern, wird beim Senden nach je 5 konsekutiven Einsen

automatisch eine Null eingefügt, ausgenommen natürlich bei den Flagbytes. Beim Empfang werden diese Nullen wieder entfernt. Diese Technik wird mit Bitstuffing bezeichnet.

- A Adresse der Unterstation (s.u.!).
- C Kontrollbyte: es enthält Kontrollnachrichten und Statusinformationen der beteiligten Stationen.
- D Datenfeld: es enthält in Form eines Bitstromes beliebiger Länge code-transparente Daten (hier: modulo-8, max. 124 Byte).
- FCS Sicherungsinformation in Form zweier CRC-Bytes, die mit dem Generatorpolynom $x^{16} + x^{12} + x^5 + 1$ erzeugt wird.

Ein Nachrichtenaustausch findet stets statt zwischen einer Hauptstation ("primary") und einer Unterstation ("secondary"). Jede Unterstation darf nur auf explizite Aufforderung der Hauptstation hin senden. Durch die Anordnung der Stationen auf einer Ringleitung bedingt, werden in jeder Unterstation eingehende Nachrichten unverändert weitergesendet ("repeater"). Wird im Adreßfeld die eigene Adresse erkannt, so wird die Nachricht in einen Puffer kopiert. Die Hauptstation empfängt dagegen alle eingehenden Nachrichten und nimmt sie vom Ring.

Um eine Kommunikation zwischen Unterstationen zu ermöglichen, wurde als Zusatzeigenschaft in der Hauptstation eine Relaisfunktion implementiert, die es erlaubt, Nachrichten von einer Unterstation auf ihren Zielort hin zu analysieren und sie gegebenenfalls an die Ziel-Unterstation weiterzugeben.

Die Kommunikationssoftware setzt sich im wesentlichen zusammen aus

(A) Basisroutinen für die
- Ein-/Ausgabe zum Arbeitsrechner,
- Ansteuerung der Leitungsinterfaces,
- Unterbrechungsbearbeitung,
- Ausgabe von Fehlernachrichten,

die gleichermaßen Verwendung finden in Haupt- und Unterstationen.

(B) Stationsspezifische Programmodule, die für Hauptstation und Unterstation unterschiedliche Struktur haben.

Die Hauptstation steuert den gesamten Kommunikationsablauf auf dem Ring. Durch ständiges, über eine "Polling"-Tabelle gesteuertes Abfragen der Unterstationen stellt sie Übertragungswünsche fest und leitet ggf. Übertragungen ein. Die daraufhin bei der Hauptstation eingehenden Nachrichten werden dort auf ihren Zielort hin analysiert. Stimmt dieser nicht mit der eigenen Adresse überein, so wird die Nachricht an die ausgewiesene Zielunterstation weitergesendet (Relaisfunktion). Der angeschlossene Arbeitsrechner ist in diesen Vorgang nicht mit einbezogen.

Im Gegensatz zur Hauptstation wird der Programmablauf der Unterstation vollständig durch Unterbrechungsanforderungen gesteuert:

- vom Leitungsinterface wird ein Unterbrechungswunsch erzeugt, wenn von der Leitung eine Nachricht empfangen wird,
- vom Arbeitsrechner wird ebenfalls eine Unterbrechung erzeugt, wenn dieser eine Anfrage oder einen Sendeauftrag an den Kommunikationsprozessor absetzen will.

Das bei der Entwicklung der Software für die Unterstation verfolgte Ziel mußte sein:

- möglichst schnelle Reaktion auf Anforderungen von der Leitung,
- möglichst schnelle Rückkehr in einen unterbrechbaren Zustand zur Vermeidung von Wartezeiten.

Die Ermittlung der Leistungsfähigkeit des Nachrichtentransportsystems ergab für die Kommunikation zwischen verteilten Prozessen über die IPC-Schnittstelle eine Übertragungskapazität von 2,1 K Byte/sec, wenn nur 2 Prozesse kommunizieren. Leistungsbegrenzend wirkt hier der Arbeitsrechner; die Grenze der Leistungsfähigkeit des Kommunikationsprozessors selbst liegt, wie andere Untersuchungen zeigten /HKK1, WEB1/, bei einem Gesamtdurchsatz von 10 K Byte/sec, wenn der Mikroprozessor 8080A zum Einsatz kommt; der Mikroprozessor vom

Typ Z80 bringt ca. 50% Leistungsverbesserung. Diese Übertragungsrate ist, bedingt durch die Masterfunktion der Hauptstation bei der jetzigen Konzeption des physikalischen Transportsystems, gleichzeitig Leistungsgrenze für die totale Datenrate (mittlere Übertragungsleistung) des gesamten Ringsystems.

Für auf Rechnern des Typs PDP11/03 residierende Prozesse folgt aus den angegebenen Leistungsgrenzen, daß die Leistungsfähigkeit des Nachrichtentransportsystems erst erschöpft ist, wenn gleichzeitig 7 logische Kanäle zwischen je zwei Prozessen mit einer Übertragungsrate von jeweils 2,1 K Byte/sec betrieben werden.

Erste Experimente mit Koordinationsprotokollen (vgl. Kap. 6) auf dem eingangs beschriebenen Pilotsystem ergaben, daß bei Einsatz des Basisprotokolls für Koordinierungsaufgaben die "konventionelle" Realisierungstechnik (Koordinatortask im Arbeitsrechner) eine mittlere Zyklusdauer von 250 Millisec. ergibt, wenn die Verweildauer im "kritischen Zustand" vernachlässigt wird.

8. Zusammenfassung

Es wurde ein Grundkonzept für die Architektur verteilter Datenbanken vorgestellt, das speziell die Einbeziehung von Kleinrechnersystemen berücksichtigt und für heterogene verteilte DV-Systeme ausgelegt ist.

Hauptmerkmal des Konzeptes ist die Einführung einer logischen Dateiebene, die die Absolvierung verteilungsspezifischer Aspekte ermöglicht und daher ein gegenüber dem Benutzer bezüglich der Verteilung der Daten transparentes Verhalten an der zugeordneten Schnittstelle anzubieten erlaubt.

Als Vorstufe zur Realisierung der logischen Dateiebene wurde ein detailliertes operationales Modell entwickelt, das auf der Schnittstelle für Interprozeßkommunikation eines Nachrichtentransportsystems für verteilte DV-Systeme basiert. Der Prototyp eines derartigen Nachrichtentransportsystems wurde im Rahmen einer Pilotimplementierung realisiert.

Die logische Dateiebene ist als Grundlage für gängige Datenbankkonzepte geeignet. Dies wurde z.B. anhand der Untersuchung der Realisierungsmöglichkeiten eines verteilten relationalen Datenbanksystems demonstriert.

Große Bedeutung kommt bei der Konzipierung und Realisierung verteilter Datenbanken der Einbettung spezieller Komponenten zu, die bei paralleler Transaktionsbearbeitung die operationale Integrität verteilter, ggf. bewußt redundant abgelegter Daten gewährleisten. Hierzu wurden geeignete Verfahren entwickelt und im Rahmen von Implementierungsstudien validiert.

Die erzielten Ergebnisse bilden die Grundlage des Folgevorhabens "Entwicklung und Implementierung von Funktionen zur Verwaltung verteilter Datenbasen".

Wesentliche Ziele dieses Vorhabens sind:

- Entwicklung der zum Aufbau der Verwaltungsinstanzen benötigten Datenmanagement-Funktionsbausteine für verteilte Systeme in weitgehend portabler Form.

- Spezifikation und Erprobung der für die Kommunikation zwischen Dateiverwaltungsinstanzen erforderlichen Kommunikationsprotokolle.
- Entwicklung von Verfahren zur Sicherung und Aufrechterhaltung der Funktion verteilter Datenbasen im Falle des Ausfalls redundanter Komponenten; Lösung des Wiederanlaufproblems;
- Einsatz dedizierter Prozessoren für Dateiverwaltungsaufgaben;
- Aufbau eines Modellierungssystems;
- Pilotimplementierung einer Datenbank mit verteilten Datenbasen.

ANHANG A

Definition der Operatoren der logischen Dateiebene

Hinweis: In eckigen Klammern stehende Parameter sind optional; bei geschweiften Klammern muß eine der angebotenen Möglichkeiten ausgewählt werden.

DML:

Satzmanipulation bei B-Dateien:

READ (<B-Datei>, <Satznummer>, <Benutzervariable>,
[<Ereignisvariable> / <Unterprogrammname>], <Return-Code>)

READS (<B-Datei>, [V / R], <Benutzervariable>,
[<Ereignisvariable> / <Unterprogrammname>], <Return-Code>)

WRITE, WRITES (Parameter s.o.)

INSERT (<B-Datei>, <Satznummer>, <Benutzervariable>,
[<Ereignisvariable> / <Unterprogrammname>], <Return-Code>)

REMOVE (<B-Datei>, <Satznummer>,
[<Ereignisvariable> / <Unterprogrammname>], <Return-Code>)

GETNAME (<B-Datei>, [<Benutzergruppe>], <Satznummer>,
<Return-Code>)

Anmerkung: Wird keine Ereignisvariable oder kein Unterprogrammname angegeben, wird der Transfer synchron abgewickelt.

SETCURSOR (<B-Datei>, {<Satznummer> / <Füllgradzeiger>}, <Return-Code>)

Satzmanipulation bei Z-Dateien

INSERT (<Z-Datei>, <Benutzervariable>, <Länge>,
[<Ereignisvariable> / <Unterprogrammname>], <Return-Code>)

REMOVE (<Z-Datei>, <Schlüssel>,
[<Ereignisvariable> / <Unterprogrammname>], <Return-Code>)

GET (<Z-Datei>, <Schlüssel>, <Benutzervariable>,
[<Ereignisvariable> / <Unterprogrammname>], <Return-Code>)

GETS (<Z-Datei>, [V / R], <Benutzervariable>,
[<Ereignisvariable> / <Unterprogrammname>], <Return-Code>)

PUT, PUTS (Parameter wie bei GET bzw. GETS und zusätzliche Länge)

SETCURSOR (<Z-Datei>, <Schlüssel>, <Return-Code>)

Dateitypunabhängige Operatoren:

WAIT (<Ereignisvariable>)

OPEN (<Log. Datei>, <Paßwort>, [SEQ], <Return-Code>)
<Paßwort> → <Paßwort für Lesen> / <Paßwort für alle Operatoren>

CLOSE (<Log. Datei>, <Return-Code>)

BEGIN_TRANS

END_TRANS

LOCK (<Log. Dateiliste>, [<Sperrtyp>], [<Satznummer>])
<Sperrtyp> → RD/WR

UNLOCK (<Log. Dateiliste>, [<Satznummer>])

CREATE (<Log. Datei>, <Return-Code>)

DELETE (<Log. Datei>, <Return-Code>)

	Meldung:	Möglich für folgende Operatoren:
<Return-Code> →	Datei nicht eröffnet	READ, READS, WRITE, WRITES, SETCURSOR, CLOSE, LOCK, UNLOCK, INSERT, REMOVE, GET, GETS, PUT, PUTS, GETNAME
	Datei nicht existent	OPEN, DELETE
	Satz nicht transferierbar	READ, READS, WRITE, WRITES, GET, GETS, PUT, PUTS
	Peripheriekapazität erschöpft	INSERT
	Benutzergruppen-Limit erreicht	INSERT, GETNAME
	Satz nicht vorhanden	READ, READS, WRITE, WRITES, SETCURSOR, GET, GETS, PUT, PUTS, REMOVE
	Verstoß gegen Schreibverbot	WRITE, WRITES, PUT, PUTS, INSERT, REMOVE, GETNAME
	falsches Paßwort	OPEN
	Transaktionsschachtelung verboten	BEGIN_TRANS
	kein Beginn der Transaktion definiert	END_TRANS
	doppeltes LOCK durch dieselbe Transaktion	LOCK
	Die Datei war nicht von der Transaktion gesperrt	UNLOCK

	Meldung:	Möglich für folgende Operatoren:
<Return-Code> →	Dateidefinition nicht vorhanden	CREATE
	nicht autorisiert	DELETE
	Satz bereits vorhanden	INSERT
	Benutzervariable nicht ausreichend dimensioniert	GET, GETS
	EOF	READS, WRITES, GETS, PUTS
	Satz/Datei nicht gesperrt	INSERT, REMOVE, WRITE, WRITES, PUT, PUTS, GETNAME

Semantik:

Transferfunktionen:

Wenn Ereignisvariable oder Unterprogramm fehlen, erfolgt der Transfer synchron. Die Benutzervariable ist ein Speicherbereich im Adreßraum des Benutzerprozesses, der einen Satz aufnimmt. Bei höheren Programmiersprachen kann dies eine Reihung oder ein Verbund sein. Die Wahl des richtigen Umfangs der Benutzervariablen ist ein Problem der Einbettung der Schnittstellenoperatoren in die Wirtssprache.

INSERT, REMOVE:

Beim Einfügen eines Satzes mit Hilfe von INSERT in eine B-Datei muß die Satznummer mit angegeben werden. Bei Z-Dateien muß ein noch nicht verwendeter Schlüssel im Satz enthalten sein. Enthält die Datei bereits einen Satz mit der angegebenen Satznummer bzw. mit dem übergebenen Schlüssel, so erfolgt eine Fehlermeldung. Die neu eingefügten Sätze können anschließend mit Hilfe von WRITE oder PUT modifiziert werden. REMOVE beseitigt einen über die Parameter spezifizierten Satz aus der angegebenen Datei.

GETNAME:

GETNAME liefert eine noch nicht verwendete Satznummer für eine B-Datei. Wird zusätzlich eine Benutzergruppe angegeben, so wird die Satznummer aus einem dieser Benutzergruppe zugeordneten Cluster beschafft. Wenn dies nicht möglich ist, erfolgt Fehlermeldung. Falls die Datei für Leser und Schreiber gesperrt ist, kann keine Satznummer beschafft werden.

OPEN:

Das Eröffnen von Dateien ist obligatorisch. Neben der Autorisierung des Benutzers wird aus dem Paßwort und der SEQ-Angabe Information für interne organisatorische Maßnahmen des verteilten Dateiverwaltungssystems gewonnen. Die SEQ-Angabe ist für Verwendung der sequentiellen Zugriffsorganisation bei B-Dateien nicht gefordert, für optimale Pufferbeschaffung jedoch erwünscht.

SETCURSOR:

Der Benutzerprozeß bekommt für alle seine eröffneten Dateien einen CURSOR. Er wird bei OPEN auf den ersten Satz der Datei gesetzt. Mit SETCURSOR wird dieser Zeiger auf den spezifizierten Satz gesetzt.
Der Füllgradzeiger ist ein pro Datei vom System geführter Zeiger, der dem Satz mit der höchsten Satznummer entspricht, der beschrieben wurde (nur für B-Datei).

READS, WRITES, GETS, PUTS:

Der Cursor wird nach Beenden des Transfers auf den nächsten Satz gesetzt.
Bei Fehlen der Richtungsangabe V bzw. R wird vorwärts gelesen.
Es können nur Sätze referenziert werden, die mit Hilfe von INSERT in die Datei eingefügt worden sind.

LOCK, UNLOCK:

Bei B-Dateien können auch einzelne Sätze gesperrt werden.

RD bedeutet sperren von Lesern und Schreibern (allgemeiner: Änderungen), WR läßt dagegen nicht verändernde Zugriffe anderer Transaktionen (READ, READS usw.) zu. Wenn bei lesenden Zugriffen keine Sperrung erfolgte, können ungültige oder inkonsistente Daten geliefert werden, bei verändernden Zugriffen dagegen müssen die entsprechenden Daten grundsätzlich über LOCK (..., RD) gesperrt worden sein.
Bei fehlendem Sperrtyp wird WR impliziert. Mit UNLOCK können einzelne Sätze nur dann freigegeben werden, wenn sie auch einzeln gesperrt worden sind.

CREATE, DELETE:

CREATE erzeugt eine Datei, deren Definition über DEF bereits katalogisiert ist.
DELETE zerstört eine Datei, wenn

- beim OPEN das für alle Operatoren autorisierende Paßwort angegeben wurde
- die Datei nicht mehr gesperrt ist.

Implizit wird neben dem Zerstören die Datei geschlossen. Die Definition der Datei bleibt hingegen weiterhin katalogisiert.

DDL:

DEF (<Dateityp>, <Log. Datei>, {<Satzlänge>,|
<Schlüsselanfang>, <Schlüssellänge>}, [<dimensionsmäßige Abschätzung des Umfangs>], <Paßwort für Leser>, <Paßwort für Leser und Schreiber>, [Quick Log. Datei>], <Sicherheitsklasse>, <Return-Code>, {<Beschreibung der Benutzergruppen>})

<Beschreibung der Benutzergruppen>
→ {{<Benutzergruppe>,}$_1^n$ <Clusterumfang>}$_1^n$ | {<Benutzergruppe>}$_1^n$

<Dateityp> → B/Z

<Benutzergruppenclusterumfang> → <Vielfaches eines Satzkontingents>

UNDEF (<Log. Datei>, <Return-Code>)

<Return-Code> →	Datei bereits definiert	DEF
	Satzlänge ≤ 0	DEF
	Benutzergruppe nicht bekannt	DEF
	Widerspruch der Abschätzung zu Benutzergruppenclusterumfang	DEF
	Katalogüberlauf	DEF
	Definition war nicht vorhanden	UNDEF

<Log. Datei> → <ASCII-Zeichen-String beliebiger Länge>

Semantik:

DEF trägt lediglich die Definition einer Datei in einen Katalog ein. Die Datei wird erst bei CREATE erzeugt. Alle Längenangaben erfolgen in Byte. Umfangabschätzungen sind Angaben über eine Anzahl von Sätzen. Bei Angabe von Benutzergruppenbeschreibungen (nur bei B-Dateien) werden entsprechende Satzintervalle den zugehörigen Benutzergruppen derart zugeordnet, daß für sie bei einem CREATE durch kompakte Speicherung physische Cluster angelegt werden.
Das Paßwort für Leser und Schreiber ermächtigt auch zur Verwendung der DELETE- und REMOVE-Operatoren. Mit Hilfe Der Sicherheitsklasse wird der Grad der Redundanz zur Erhöhung der Verfügbarkeit einer zu erzeugenden Datei festgelegt.
UNDEF entkatalogisiert die Dateibeschreibung, vorausgesetzt, daß die Datei noch nicht bzw. nicht mehr existiert.

MDL:

MAP (<Log. Datei>, [<VG-Dateien-Liste>], <Return-Code>)

Es kann auf MAP und die Spezifikation der Objekte der nächst tieferen Schicht verzichtet werden. Die Namen der Objekte werden dann automatisch erzeugt. MAP dient dazu, diesen automatischen Abbildungsvorgang außer Kraft zu setzen.

Weiterhin stehen dem Datenbasisadministrator weitere Funktionen und Dienstprogramme zur Verfügung, die jedoch nicht un-

mittelbar auf der Ebene der logischen Dateien angesiedelt werden können und daher hier nicht definiert werden. Zu den Dienstprogrammen gehören Programme zum Ausdrucken von Katalog-informationen und Übersichten, Kopieren von Dateien, Laden von Z-Dateien, Programme zur Erstellung von Statistiken, zum Anlegen von Sicherungsbeständen, Auswerten von Log-Bändern, Ändern von Attributen wie z.B. Paßworte etc.

Definition der Operatoren der G'-Ebene

DML:

READ (<G'-Datei>, <Blocknr.>, <Benutzervariable>,
{<Ereignisvariable> / <Unterprogrammadr.>}, <Return-Code>)

WRITE (Parameter wie bei READ)
<G'-Datei> → <ASCII-String aus 6 Zeichen>

WAIT (<Ereignisvariable>)

OPEN (<G'-Datei>, <Paßwort>, <Return-Code>)

CLOSE (<G'-Datei>, <Return-Code>)

CREATE/DELETE (<G'-Datei>, <Return-Code>)

LOCK, UNLOCK (<G'-Datei>, {RD / WR})

<Return-Code> →	G'-Datei nicht eröffnet	READ/WRITE/CLOSE
	Blocknr. zu groß	READ/WRITE
	Blocknr. < 0	READ/WRITE
	Block nicht lesbar	READ/WRITE
	Verstoß gegen Schreib-verbot	WRITE
	falsches Paßwort	OPEN
	nicht lesbarer Daten-träger	OPEN
	G'-Datei nicht vor-handen	OPEN
	Datei bereits eröffnet	OPEN
	nicht autorisiert	DELETE

DDL:

DEF (<G'-Datei>, <Längenfaktor>, <Paßwort für Leser>,
<Paßwort für Leser und Schreiber>, [Quick
<G'-Dateiname>], <Return-Code>)

Anmerkung: Das Leser-Schreiber-Paßwort autorisiert zu DELETE-Aufruf

UNDEF (<G'-Datei>, <Return-Code>)

<Return-Code> →	G'-Datei bereits definiert	DEF
	Längenfaktor ≤ 0	DEF
	Katalogüberlauf	DEF
	G'-Datei nicht definiert	UNDEF

Der Längenfaktor definiert eine Anzahl von Blöcken, die den Dateiumfang festlegen. Die Quick-Option kann weggelassen werden. Im Falle ihrer Verwendung muß die darin angegebene Datei nicht unbedingt bereits existieren. Sie bewirkt abhängig von der Angabe einer eventuell vorhandenen MAP-Anweisung eine physisch benachbarte Unterbringung einer Datei oder Unterbringung auf einem simultan zugreifbaren Datenträger (parallele Ablage).

MDL:

MAP (<G'-Datei>, <L-Datei>, {<Datenträger> / <LUN>}, <Return-Code>)

<Return-Code> →	G'-Datei nicht definiert
	L-Datei nicht definiert
	Datenträger nicht vorhanden
	LUN nicht vorhanden

Anmerkung: Mit Hilfe der MAP-Anweisung kann der Datenbasisadministrator die Plazierung der G'-Datei folgendermaßen steuern:

- Rechner ist bereits definiert, da der Adreßraum für G'-Dateien die Datenträger des (DISKPACKS, DISKS) lokalen Rechners sind

- Unterbringung auf einem speziellen Datenträger
- Unterbringung auf einem auf einer LUN (Logischen Gerät) gerade montierten Datenträger
- auf irgendeinem montierten Datenträger, auf dem eine L-Datei mit dem angegebenen Namen eingerichtet ist
- durch Variation des L-Dateinamens ist nochmals eine unterschiedliche Plazierung innerhalb eines Datenträgers möglich.

ANHANG B

Verzeichnis der mit dem Vorhaben "Datenbankarchitektur für verteilte DV-Systeme" in Zusammenhang stehenden Veröffentlichungen, Tagungsbeiträge, Seminarveranstaltungen

Veröffentlichungen

Drobnik, O.: Strukturmodelle dezentralisierter Kontrolle in Mehrrechnersystemen, NTG-GI-Fachtagung "Rechnernetze und Datenfernverarbeitung", Aachen 1976, Informatik Fachberichte 3, Springer Verlag

Drobnik, O.: Verfahren zur Sicherung der operationalen Integrität in verteilten Datenbasen bei dezentraler Kontrollstruktur, Dissertation, Universität Karlsruhe, 1977

Holler, E., Drobnik, O.: Integrität, Ausfall und Wiederanlauf redundanter Prozeßdatenbasen in verteilten PDV-Systemen, GMR-GI-GfK-Fachtagung Prozeßrechner, Augsburg, März 1977, Informatik Fachberichte 7, Springer Verlag

Holler, E., Krieger, J., Knöpker, R.: Ein universeller Kommunikationsprozessor für den Aufbau verteilter PDV-Systeme, GMR-GI-GfK-Fachtagung Prozeßrechner, Augsburg, März 1977, Informatik Fachberichte 7, Springer Verlag

Holler, E., Drobnik, O.: Implementation of decentralized coordination mechanisms in distributed mini-/ microcomputer systems, "Distributed Processing Workshop", Brown University, Providence-Rhode Island, 1977

Krieger, J., Knöpker, R.: Realisierung von Kommunikationsfunktionen mit Mikroprozessoren, PDV-Fachtagung: "Einsatz von Mikroprozessoren zur Prozeßlenkung", Karlsruhe 3.-4. Nov. 1976, KFK-PDV 101, Jan. 1977

Wolfinger, B., Drobnik, O., Holler, E.: Design and Simulation of Decentralized Control Structures for Reliable Distributed Systems, Third Winterschool on the Theory of Operating Systems, Visegrád, 1977 (Conf. Proc. werden erscheinen)

Diplomarbeiten

Moster, H.J.: Lösung des "Reader-Writer-Problems" für Datenbasen in Rechnernetzen, Diplomarbeit, Universität Karlsruhe, 1976

Weber, E.: Simulation der Kommunikation verteilter Rechner über ein Ringleitungssystem, Diplomarbeit, Universität Karlsruhe, 1977

Vorträge

Breitwieser, H., Drobnik, O., Holler, E., Keil, C., Knöpker, R., Krieger, J.: Präsentation des DISCO-Projekts (Distributed Data Bases for Small Computers) im Rahmen des Arbeitskreises "Verteilte Datenbanken" im IDT des Kernforschungszentrums Karlsruhe, Jan. 1977

Breitwieser, H., Drobnik, O., Holler, E., Keil, C., Kersten, U.: Presentation of the DISCO-Project, French-German-Seminar on Distributed Databases, IDT-Nuclear Research Center Karlsruhe, Oct. 1977

Holler, E.: Transaction Control in Distributed Data Base Systems, präsentiert im Rahmen einer Forschungsreise in die USA und Kanada an den Universitäten von Washington in Seattle und Waterloo in Ontario sowie am IBM Research Center in Yorktown, 1976

Holler, E., Drobnik, O.: Implementierung von dezentralen Koordinationsmechanismen in Kleinrechnernetzen, MIMI, Zürich, Juni 1977

Holler, E., Keil, C.: Präsentation des DISCO-Projektes beim Battelle-Institut in Frankfurt, Febr. 1977

Seminare

Arbeitskreis "Verteilte Datenbanken": Ausrichtung eines Arbeitskreistreffens im IDT des Kernforschungszentrums Karlsruhe am 31. Jan. 1977

French-German-Seminar on "Distributed Databases": Ausrichtung des Seminars im IDT des Kernforschungszentrums Karlsruhe am 25.-26. Okt. 1977

Literatur

/ABS1/ Akkoyunlu, E., Bernstein, A., Schantz, R.: Interprocess Communication Facilities for Network Operating Systems, Computer, June 74, pp. 46-55

/ANS1/ ANSI/X3/SPARC Study Group on Data Base Management Systems, Interim Report 1975, Doc.No. 7514TS01

/AUE1/ Auerbach: Computer Technology Reports Digital Equipment - Digital Network Architecture and DDCMP

/BAM1/ Bayer, R., McCreight, E.: Organization and Maintenance of Large Ordered Indexes, Acta Informatica 1, 1972, pp. 173-189

/BAY1/ Bayer, R.: Integrity, concurrency, and recovery in databases, ECI 1976, Springer Verlag, Lecture Notes in Computer Science 44, pp. 81-106

/BER1/ Berg, J.L.: Data Base Directions - The Next Steps, Proc. of the Workshop of the National Bureau of Standards and the Association for Computing Machinery, Fort Landerdale, Florida, Oct. 29-31, 1975

/CAS1/ Casey, R.G.: Allocation of copies of a file in an information network, Spring Joint Computer Conference, 1972

/CEK1/ Cerf, V.G., Kahn, R.E.: A proposal for Packet Network Intercommunication, IEEE Transactions on Communications COM-22, No. 5, May 1974

/CER1/ Cerf, V.G. et al.: Proposal for an International End-to-End Protocol, SIGCOMM, Vol. 6, No. 1, Jan. 1976, pp. 63-89

/CHA1/ Champine, G.A.: Six Approaches to Distributed Data Bases, Datamation, May 1977, pp. 69-72

/CCI1/ CCITT Study Group VII: Draft Recommendation X.25, Sections 1-5, Temporary Document No. 36-E, 37-E, March 1976

/CHM1/ Crocker, S.D., Heafner, J.F., Metcalf, R.M., Postel, J.B.: Function-Oriented Protocols for the ARPA Computer Network, Spring Joint Computer Conference, 1972

/CHS1/ Chupin, J.-C., Seguin, J.: A Network Direkt Access Method, Proc. European Workshop on Distributed Computer Systems, Darmstadt, 1974

/CHU1/ Chu, W.W.: Performance of file directory systems for data bases in star and distributed networks, National Computer Conference, 1976

/CHU2/ Chupin, J.-C.: Control Concepts of a Logical Network Machine for Data Banks, Information Processing 74, North-Holland Pub. Comp., 1974

/CHU3/ Chu, W.W.: Optimal File Allocation in a Multiple Computer System, IEEE Transactions on Computers, No. 10, Oct. 1969, pp. 885-889

/CJM1/ Cosell, B.P., Johnson, P.R., Malman, J.H., Schantz, R.E., Sussman, J., Thomas, R.H., Walden, D.C.: An Operational System for Computer Resource Sharing, Bolt Beranek and Newman Inc., 1975

/COD1/ CODASYL Data Base Task Group Report, April 1971, erhältlich bei: IFIP Data Processing Group

/COD2/ Codd, E.F.: A Relational Model of Data for Large Shared Data Banks, CACM, Vol. 13, No. 6, 1970, pp. 377-387

/COD3/ Codd, E.F.: Relational Completeness of Data Base Sublanguages, in Courant Computer Science Symposion, Vol. 6: "Data Base Systems", edited by Rustin R., Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1971, pp. 65-98

/DAB1/ Danthine, A., Bremer, J.: Définition, représentation et simulation de protocoles dans une contexte réseaux, International Meeting on Mini-Computers and Data Communication, Liège, Jan. 20-22, 1975

/DEC1/ Digital Equipment Corporation: DECNET-Design Specifications for: Data Access Protocol (DAP), July 1975

/DIT1/ Dittmann, E.L.: Klassifizierung von Datenunabhängigkeit für den Systementwurf, TH Darmstadt, Bericht der Informatik-Forschungsgruppe DV75-1, 1975

/DRO1/ Drobnik, O.: Verfahren zur Sicherung der operationalen Integrität in verteilten Datenbanken bei dezentraler Kontrollstruktur, Dissertation, Universität Karlsruhe, 1977

/ESW1/ Eswaran, K.P.: Placement of records in a file and file allocation in a computer network, IFIP Conference Proc., Stockholm, 1974, pp. 304-307

/ESW2/ Eswaran, K.P. et al.: On the notions of consistency and predicate locks in a data base system, CACM, Vol. 19, No. 11, Nov. 1976, pp. 624-633

/FAL1/ Falkenberg, E.: Concepts for the coexistence approach to data base management, Proc. of the International Computing Symposium, Liège, Belgium, April 1977, North-Holland Publishing Company, Amsterdam 1977

/GHO1/ Ghosh, S.P.: Distributing a Data Base with Logical Associations on a Computer Network for Parallel Searching, IEEE Transactions on Software Engineering, No. 6, 1976, pp. 106-113

/GJP1/ Gardarin, G.,Jouwe, M., Parent, C., Spaccapietra, S.: Designing a Distributed Data Base Management System, AICA 77, Pisa, 12-14 Okt. 1977

/GOT1/ Gotlieb, L.R.: Computing Joins of Relations, Proc. of the ACM SIGMOD Conference, 1975

/GRA1/ Gray, J.N. et al.: Granularity of locks and degrees of consistency in a shared data base, GMD-Sommerseminar über Datenbanktechnologie, Bonn-St. Augustin, Juli 1976

/HAE1/ Härder, Th.: An Implementation Technique for Generalized Access Path Structure, IBM RJ 1837, 1976

/HEB1/ Heinze, W., Butscher, B.: File Transfer in the HMI-Computer-Network, 1976, unveröffentlicht

/HEI1/ Heinrich, F.R.: The Structure of a Distributed Computing System - the Distributed File System, University of California, Irvine, Information and Computer Science Dept., 1972

/HKK1/ Holler, E., Krieger, J., Knöpker, R.: Ein universeller Kommunikationsprozessor für den Aufbau verteilter PDV-Systeme, GMR-GI-GfK-Fachtagung: Prozeßrechner 1977, Augsburg, 7. u. 8. März 1977, Tagungsband Informatik Fachberichte 7, Springer Verlag

/HOD1/ Holler, E., Drobnik, O.: Integrität, Ausfall und Wiederanlauf redundanter Prozeßdatenbasen in verteilten PDV-Systemen, 2. Fachtagung "Prozeßrechner 1977", Augsburg, März 1977

/HOD2/ Holler, E., Drobnik, O.: Rechnernetze, Reihe Informatik (17), BI Wissenschaftsverlag, Zürich 1975

/HOL1/ Holler, E.: Koordination kritischer Zugriffe auf verteilte Datenbanken in Rechnernetzen bei dezentraler Überwachung, Dissertation, Universität Karlsruhe 1974

/HOL2/ Holler, E.: Multiple Copy Files in Computer Networks, KFK-Bericht 1734, 1974

/HPT1/ Hewlett Packard: Tech. Notes, Distributed Systems Technical Data 5952-1645, 1975

/INT1/ High Level Data Link Control Procedures Proposed Draft International Standard on Elements of

Procedures, International Organization for Standardization ISO/TC97/SC6, Mai 1975

/INT2/ INTEL 8080 Microcomputer Systems Users' Manual, Sept. 1975

/JOT1/ Johnson, P.R., Thomas, R.H.: The Maintenance of Duplicate Databases, Network Working Group, RFC # 677, NIC # 31507, 1975

/KAR1/ Karl, H.: The Distributed Data Bases of the Information System of the German Police (INPOL), European Computer Workshop Series, Distributed Computer Systems, Darmstadt, Germany, Oct. 1974

/KRK1/ Krieger, J., Knöpker, R.: Realisierung von Kommunikationsfunktionen mit Mikroprozessoren, PDV-Fachtagung: "Einsatz von Mikroprozessoren zur Prozeßlenkung", Karlsruhe 3.-4. Nov. 1976, KFK-PDV 101, Jan. 1977

/LAC1/ Lagasse, J.P., Artaud, G., Cabanel, J.P.: ARAMIS - A Processing Network with User Data Bases Interactive Systems, COMPCON, 1975

/LEM1/ Le Moli: A Theory of Colloquies, IRIA Proc. of the 1st European Workshop on Computer Networks, Arles, 1973

/LEM2/ Levin, K.D., Morgan, H.L.: A Dynamic Optimization Model for Distributed Databases, Report 76-09-01, Univ. of Pennsylvania, Philadelphia, 1976

/MAR1/ Mahmoud, R.: Optimal Allocation of Resources in Distributed Information Networks, ACM-TODS, No. 1, 1976

/MAS1/ Marill, T., Stern, D.: The datacomputer - A network data utility, National Computer Conference, 1975

/MCG1/ McGee, W.C.: The information management system IMS/VS, IBM Systems Journal, Vol. 16, No. 2, 1977, pp. 84-168

/MIL1/ Mills, D.L.: Dynamic File Access in a Distributed Computer Network, University of Maryland, TR-415, 1975

/NBC1/ Nahouraii, E., Brooks, L.O., Cardenas, A.F.: An approach to data communication between different generalized data base management systems, 2nd International Conference on Very Large Data Bases, Brüssel, Sept. 1976

/NIJ1/ Nijssen, G.M.: A gross architecture for the next generation Database Management System, Proc. IFIP TC2 Working Conference on "Modelling in Database Management Systems", Freudenstadt, Germany, Jan. 1976, North-Holland, Publishing-Company, Amsterdam, 1976, pp. 1-24

/PAT1/ Patz, M.: Entwurf von Kommunikationskontrollern mit Mikroprozessoren, PDV-Fachtagung: "Einsatz von Mikroprozessoren zur Prozeßlenkung", Karlsruhe, 3.-4. Nov. 1976, KFK-PDV 101, Jan. 1977

/PEE1/ Peebles, R.W.: Design Considerations for a Distributed Data Access System, Ph. D. dissertation, University of Pennsylvania, Aug. 1972

/SAA1/ Senko, M.E., Altman, E.B., Astrahan, M.M., Fehder, P.L.: Data Structures and Accessing in Data Base Systems, IBM Systems Journal, Vol. 12, No. 1, 1973, pp. 30-93

/SAB1/ Sabinowski, H.: PIX Overview, The Third European Network Users' Workshop, IIASA, Laxenburg, Austria, 19/20 April 1977

/SCB1/ Schmid, H.A., Bernstein, P.: A multi-level architecture for relational data base systems, Proc. of the Boston Conference on Very Large Data Bases, 1975

/SCH1/ Schreiber, F.A.: Distributed Data Bases: Some problems still to be solved, Proc. of Convention Informatique 1975, Paris, 1975

/SCH2/ Schreiber, F.A.: A framework for distributed data base systems, Proc. of the International Computing Symposium, Liège, Belgium, April 1977, North-Holland Publishing Company, Amsterdam, 1977

/SCH3/ Schneider, L.S.: A Relational Query Compiler for Distributed Heterogeneous Databases, 1976, unveröffentlicht

/SCH4/ Schlageter, G.: Konsistenzbedingungen für Datenbanksysteme: Ein neues Problem der Systemanalyse, Angewandte Informatik, Vol. 4, 1976, pp. 159-162

/SCH5/ Schmid, H.A.: Architektur und Implementierung von Datenbanksystemen, GMD-Sommerseminar über Datenbanktechnologie, Bonn-St. Augustin, Juli 1976

/SHI1/ Shima, M. et al.: Z-80 Chip set herolds third microprocessor generation, Electronics, 19. Aug. 1976, pp. 89-93

/SLH1/ Shu, N.C., Lum, V.Y., Housel, B.C.: An Approach to Data Migration in Computer Networks, Proc. of the Berkeley Workshop on Distributed Data Management and Computer Networks, May 1976

/SMC1/ Smith, J.M., Chang, P.Y.-T.: Optimizing the Performance of a Relational Algebra Database Interface, CACM, Vol. 18, No. 10, 1975, pp. 568-579

/SPT1/ Stygar, P., Puchrik, A., Turek, M.: Transparent Integrated Intelligence Network Query Intermediate Processor, Rome Air Development Center (IRDA), Final Technical Report RADC-TR-77-39, Jan. 1977

/STN1/ Stonebraker, M., Neuhold, E.: A Distributed Data Base Version of INGRES, Engineering Research Laboratory, University of California, Berkeley, Memorandum No. ERL-M612, Sept. 1976

/STO1/ Stonebraker, M.: Proposal For a Network INGRES, Proc. of the Berkeley Workshop on Distributed Data Management and Computer Networks, May 1976

/STO2/ Stonebraker, M.: A Comparison of Links and Secondary Indexes in a Relational Data Base System, Proc. of the IEEE Conference on Software Engineering, San Francisco, Ca., Oct. 1976

/STS1/ Strack-Zimmermann, H.W., Schrödter, H.D.: The Hahn-Meitner-Institut Computer Network, NTG-GI-Fachtagung "Rechnernetze und Datenfernverarbeitung" Aachen 1976, Informatik Fachberichte 3, Springer-Verlag 1976

/TOT1/ Totaro, J.B.: Communications Processor Survey, Datamation, Mai 1976, pp. 151-170

/TSI1/ Tsichritzis, D.: A Network Framework for Relational Implementation, University of Toronto, Computer Systems Research Group Report CSRG-51, 1975

/WAL1/ Walden, D.C.: A System for Interprocess Communication in a Resource Sharing Computer Network, CACM, Vol. 15, No. 4, April 1972, pp. 221-230

/WEB1/ Weber, E.: Simulation der Kommunikation verteilter Rechner über ein Ringleitungssystem, Diplomarbeit an der Fakultät für Informatik der Universität Karlsruhe, 1977

/WHI1/ Whitney, V.K.M.: A Study of optimal file assignment and communication network configuration in remote-access computer message processing and communication systems, SEL Technical Report No. 48, University of Michigan, Ann Arbor, 1970