KONZEPT EINES ALLGEMEINEN BIBLIOTHEKSVERBUNDSYSTEMS

von

H. Langendörfer und H. Scheidig

Bericht Nr. A 78-20

Fachbereich 10
Angewandte Mathematik und Informatik
Universität des Saarlandes

D-6600 Saarbrücken

November 1978

Inhaltsverzeichnis

Einleitung:

Seit geraumer Zeit werden in Bibliotheken gewisse, immer wiederkehrende Arbeitsgänge mit Hilfe der Datenverarbeitung automatisiert. Es handelt sich dabei durchweg nur um die Automatisierung einzelner Funktionen im Rahmen bibliothekarischer Buchbearbeitung.

Heute plant man die Entwicklung von (Programm) Systemen, die die Integration verschiedener Arbeitsschritte mehrerer angeschlossener Bibliotheken in einem Verbund vorsehen.

Ziel derartiger Vorhaben ist sowohl ein regionaler Bearbeitungsverbund als auch die Einbettung der Bibliotheken in nationale und internationale Informationsnetze.

Im folgenden wird das Konzept für die Software eines solchen Bibliothekenverbundes entwickelt. Beim Entwurf des Systems wurde nach der top (benutzer- bzw. problemorientierten)- dow (maschinennah, programmierorientiert) Strategie vorgegangen.

Systemanforderungen:

Leistungsumfang:

1) Die für On-line Verbundsysteme relevanten Bibliotheksbereiche sind die
   - Katalogisierung
   - Sacherschließung
   - Erwerbung
   - Benutzung

   und das Rechnungswesen.

Eigenschaften des Systems:

2) Es muß die Portabilität der Programme zwischen den einzelnen Bibliotheksregionen möglich sein.

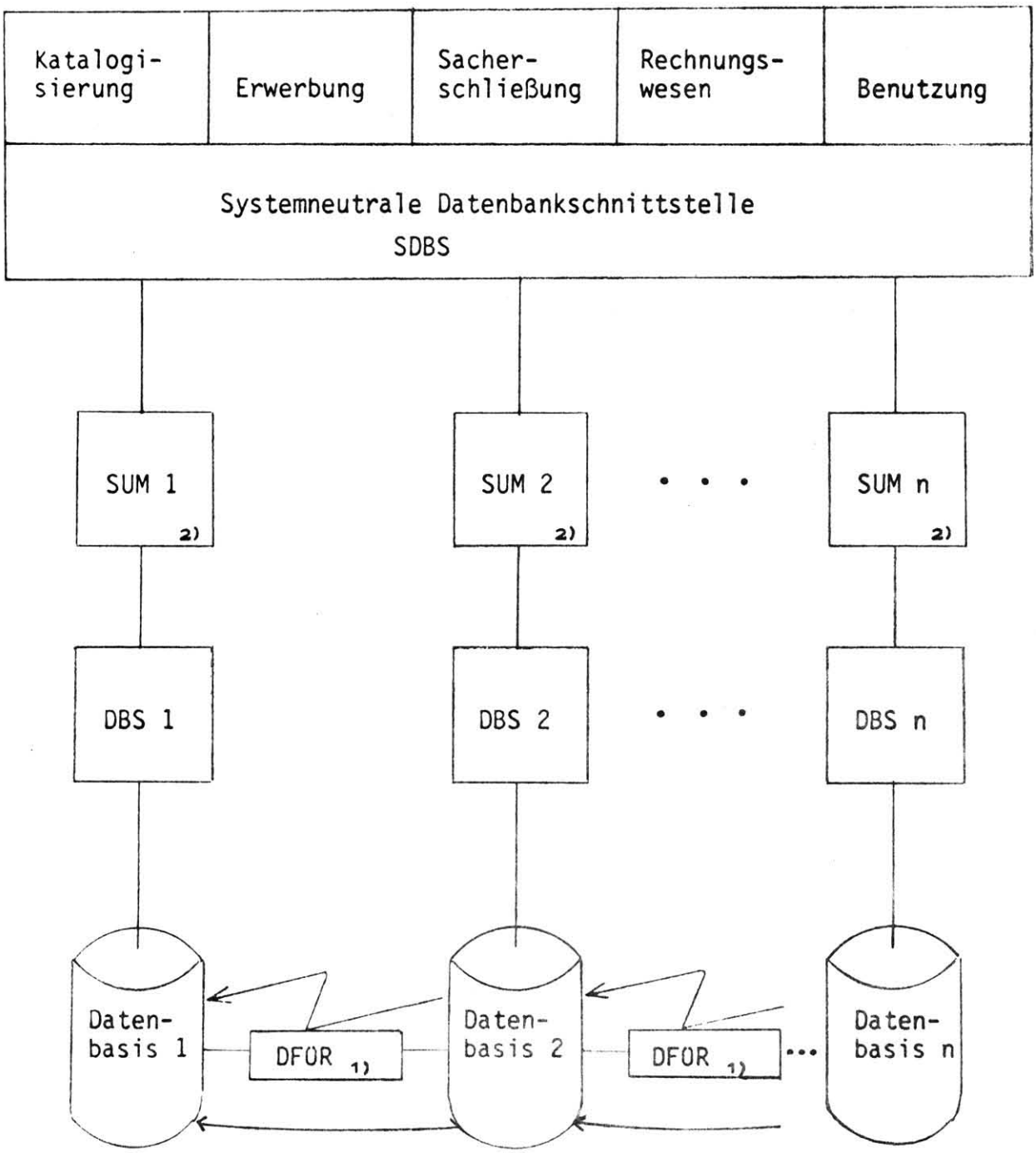
3) Das System muß der Integration der Verbunde in bestehende/geplante nationale/internationale Informationsnetze Rechnung tragen sowie der Vernetzung der Verbunde untereinander.

4) Die Kontinuität der Geschäftsabläufe in den einzelnen dem Verbund angehörenden Bibliotheken muß bei evtl. Hardwareänderungen gewährleistet sein (Migrationsaspekt).

Eine Analyse der Bearbeitungsvorgänge Katalogisierung, Sacherschließung usw. zeigt, daß diese immer wieder die gleichen Grundfunktionen wie Erfassen, Suchen, Ändern und Löschen von Daten benutzen. Dies zeigt, daß sich die Komponenten Katalogisierung, Erwerbung usw. auf ein geeignetes Datenverwaltungssystem abstützen müssen (daneben auch wegen der Datenunabhängigkeit der Programme).

Nun erlaubt aber aufrund der regional verschiedenen Hard- und Software, insbesondere auch der Datenbanksoftware, lediglich eine systemneutrale und kompatible Datenbankschnittstelle übertragbare und damit überregional einsetzbare Programme zu erstellen. Eine Umsetzung dieser Schnittstelle auf die gegebenen Datenverwaltungssysteme muß vorgesehen werden. Die Integration der Bibliotheksverbunde in Informationsnetze einerseits und die ins Auge gefaßte Vernetzung der Verbunde untereinander ist nur möglich, wenn das System u.A. eine geeignete Daten(bank)-übertragungskomponente enthält. Diese muß auch die Kontinuität der Geschäftsabläufe bei evtl. Hardwareänderungen gewährleisten. Ebenso muß sie die Restrukturierung bereits vorhandener Datenbanken ermöglichen.

Damit liegen zunächst einmal die Komponenten fest. Das folgende Schema gibt eine Übersicht über die Komponenten, sowie über die Architektur des Systems:



1) Datenfernübertragungs bzw. Datenbank-Restrukturierungs-Komponente

2) Schnittstellenumsetzer

I. Komponentenanforderungen:

Die Anforderungen an die Komponenten Katalogisierung, Erwerbung, Sacherschließung usw. ergeben sich im wesentlichen aus den bibliothekarischen Bearbeitungsrichtlinien [KOHL 78]. Hierauf wird an dieser Stelle nicht näher eingegangen. Näher eingegangen wird nur auf die Komponenten DFÜ, DBS bzw. SDBS.

1. Anforderungen an Datenverwaltungssysteme bzw. an SDBS

Diese Anforderungen ergeben sich aus der Beschaffenheit der bibliographischen Daten und den Forderungen für ihre Verarbeitung in einem Bibliotheksverbund.

1.1 Aufbau bibliographischer Datensätze

Die bibliographischen Datensätze sind Sätze variabler Länge mit einer variablen Anzahl von Feldern. Jedes Feld wiederum kann fester oder variabler Länge sein.

1.2 Satztypen

Im folgenden werden Satztypen aufgezeigt, die in einer bibliothekarischen Datenbank auftreten

- Titelsätze
  Dabei werden unterschieden

  Hauptsätze = Datensätze für Monographien, Serien, etc.

  Nachsätze = Datensätze für spezielle Nebeneintragungen

  Untersätze = Datensätze für Teile einer mehrbändigen Veröffentlichung.

- Personensätze

  Datensätze für Ansetzungs- und Verweisformen von Personennamen.

- Körperschaftssätze

  Datensätze für Ansetzungs- und Verweisformen von Körperschaftsnamen.

- Lokale Datensätze

  In einem Verbundsystem Datensätze für bibliotheks-individuelle Daten, Bearbeitungszustände, Signatur, etc.

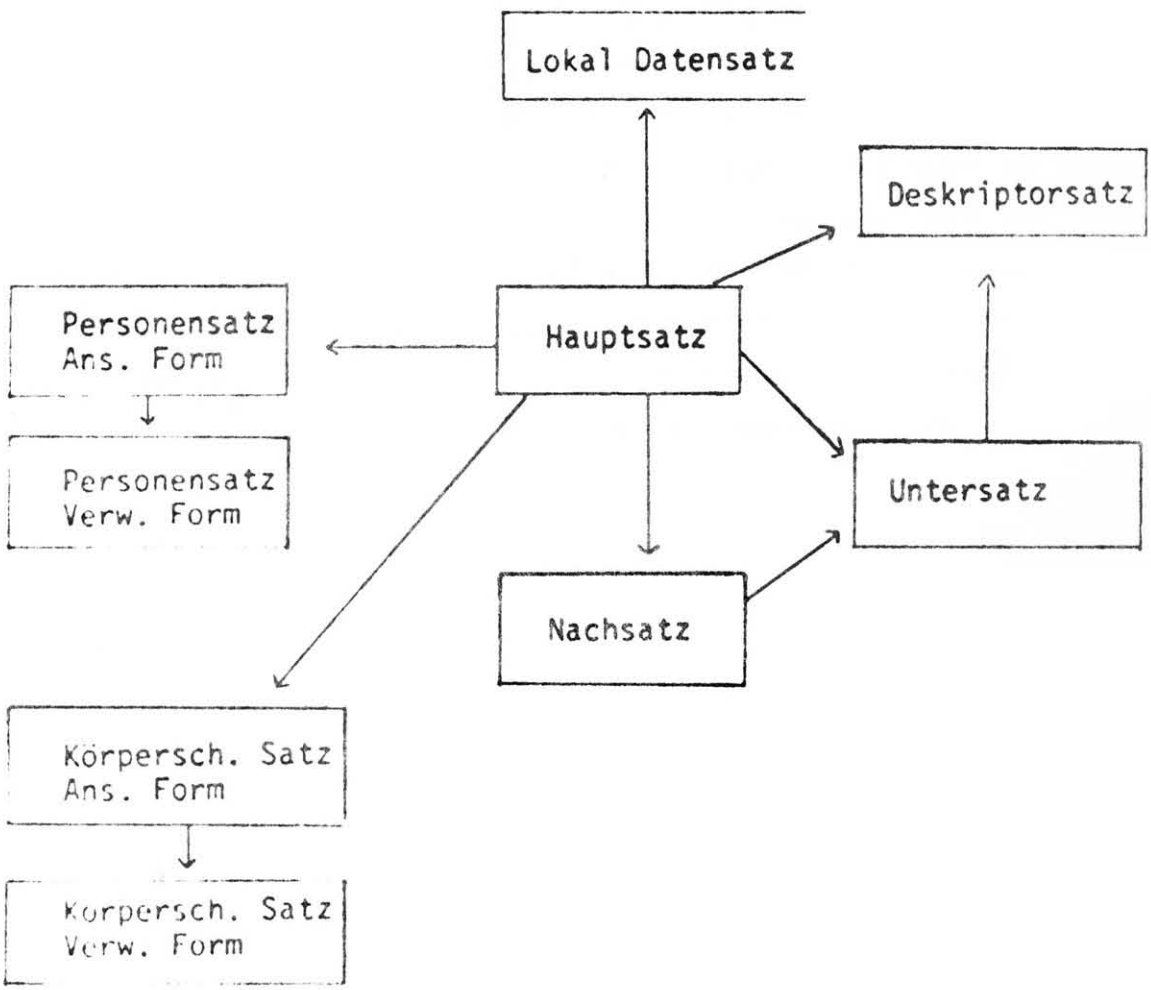
- Deskriptorensätze

  Datensätze für Schlagworte, Suchbegriffe, etc.

1.3 Datenbankstruktur

Der in der nachstehenden Abbildung gezeigte Datenbakstrukturplan stellt einen Ausschnitt aus einer bibliothekarischen Datenbank (Katalogisierung mit Sacherschließung) dar und zeigt die möglichen Satzbeziehungen auf.

Datenbankstrukturplan
(Katalogisierung mit Sacherschließung)



Dieser (Teil-) Datenbankstrukturplan zeigt, daß das zugrundegelegte Datenverwaltungssystem ein Netzwerk-Modell sein muß.

1.4 Datenmanipulation

Zur Überprüfung und Erhaltung der semantischen Integrität der Daten müssen Integritätsbedingungen formuliert werden können. Sie beschreiben einen korrekten Zustand der Datenbank und zulässige Übergänge von einem Zustand der Datenbank in einen anderen.

Das Datenbaksystem muß Anweisungen, die die Vergabe von Zugriffsrechten auf Datenobjekte ermöglichen, enthalten. Diese Berechtigungs-Anweisungen sind notwendig, um eine zentrale redaktionelle Verantwortung für den Datenbestand zu gewährleisten.

1.5 Datenwiedergewinnung und Zugriffsmechanismus

Es muß möglich sein, benutzerspezifische Zugriffspfade einzurichten und zu pflegen und wieder zu zerstören. Die Datenbank muß in Bereiche eingeteilt werden können, entsprechend den Benutzerwünschen. Auf den Bereichen müssen Zugriffspfade eingerichtet werden können. Dies sind im wesentlichen die Anforderungen soweit sie aus den bibliothekarischen Daten und aus der Verarbeitung unter Verbundaspekten heraus gegeben sind. Auf die allgemeinen Forderungen an ein Datenverwaltungssystem braucht an dieser Stelle nicht mehr eingegangen zu werden. Gleichzeitig muß aber bei dem ungeheuren Datenanfall gewährleistet sein, daß für die häufigen Benutzungen auch auf logischer Ebene, also sichtbare, Zugriffspfade zur Verfügung stehen, weil sonst das Leistungsverhalten des Systems schlecht und damit die Akzeptanz geringer wird.

2. Anforderungen an die Komponente DFÜR

Die allgemeinen Anforderungen an diese Komponente wurden in der Arbeit "Stored Data Description and Data Translation: A Model and Language" Information Systems 3/1977 von der Stored-Data Definition and Translation Task Group of the CODASYL-Systems Committee beschrieben. [FRY 77]. An dieser Stelle wird nur noch kurz darauf eingegangen.

2.1 Beschreibungssprache für gespeicherte Daten (SDDL-Stored Data Description Language)

Ein wichtiger Bestandteil dieser Komponente ist die Datenbeschreibungssprache für gespeicherte Daten. Diese muß die Beschreibung

- der logischen Struktur und ihre Codierung

- der physikalischen Struktur der Datenträger

und

- die Abbildung von der logischen auf die physikalische Struktur

ermöglichen. Hierzu ist u.A.ein Datenmodell erforderlich, das alle Modelle gespeicherter Daten umfaßt, sowie die Unabhängigkeit der logischen Beschreibung der Daten von der physikalischen Datenbeschreibung gewährleistet.

2.2 Datenübertragungssprache (TDL-Translation Description Language)

Die Komponente muß eine Sprache enthalten, in der die Beziehungen zwischen Quell- und Zieldaten für die eigentliche Übertragung beschrieben werden.

2.3 Übertragungsprozessor

Bestandteil dieser Komponente muß ein Übertragungsprozessor sein für die Konvertierung der Daten von einer physikalischen und logischen Struktur in die andere, einschließlich der Algorithmen, interner Datendarstellung und Optimierungstechniken.

3. Anforderungen an die Komponenten SUM

Die Anforderungen an diese Komponenten werden im wesentlichen durch das zugrundegelegte Datenbanksystem durch das die Allgemeine Schnittstelle realisiert werden soll, gegeben. Sie können im einzelnen hier noch nicht angegeben werden.

II. Beschreibung der einzelnen Komponenten:

In diesem Teil werden die einzelnen Komponenten, soweit sie durch die Anforderungen festliegen, beschrieben.

1. Realisierung von SDBS durch ELSL (Extended Link and Selektor Language)

ELSL ist eine dialogorientierte Datenbanksprache zur Definition, zur Manipulation und für das Retrieval von Daten einer Datenbank. ELSL ist für die parallele Bearbeitung von Datenbanken durch mehrere Benutzer ausgelegt. Von einem Benutzerprozeß können mehrere Datenbanken gleichzeitig bearbeitet werden.

Zur Definition von Datenstrukturen bietet ELSL die Möglichkeiten des Relationenmodells, des Hierarchischen Modells und des Netzwerkmodells, wobei die "manual links" des Hierarchischen bzw. Netzwerkmodells durch die "automatic links" in ELSL ersetzt werden müssen. ELSL erlaubt auf logischer Ebene die Definition von Zugriffspfaden (Invertierung, Selektoren, Links). Diese logischen Zugriffspfade ermöglichen - bei entsprechender physikalischer Realisierung - einen schnellen Zugriff auf bestimmte Teile einer Datenbank.

ELSL erlaubt sehr differenzierte Sperren auf Datenobjekten; unter anderem können auch prädikative Sperren formuliert werden. Ein Benutzer muß bei einem Zugriff auf die Datenbank nur die wirklich benötigten Teile der Datenbank sperren.

ELSL enthält Sprachelemente zur Sicherstellung der semantischen Integrität einer Datenbank. Der Datenbankadministrator kann Zugriffsrechte für alle Arten von Operationen auf (Teilen von) Datenobjekten vergeben. Insbesondere durch die Unterscheidung zwischen *read*- und *output*-Recht wird ein größerer Datenschutz gewährleistet. Die Informationsfunktionen in ELSL stellen eine notwendige Ergänzung für eine Datenbanksprache dar.

Der Retrieval-Teil von ELSL lehnt sich eng an die Relationenalgebra an. Er ist vollständig, d.h. in ELSL können alle Operationen der Relationenalgebra formuliert werden.

ELSL ist für die konzeptuelle Ebene eines Datenbanksystems entworfen. Um ELSL in ein funktionsfähiges Datenbanksystem zu integrieren, ist eine Realisierung der ELSL-Datenstrukturen auf interner Ebene z.B. durch virtuelle Dateien notwendig. Zur Festlegung der physikalischen

Ablageform dieser virtuellen Dateien sind dann auf interner Ebene zusätzliche Sprachmittel notwendig.

ELSL ist in der in [ASCHENBRENNER 78] beschriebenen Form eine reine Dialogsprache, kann aber mit geringem Aufwand in eine Host-Language eingebettet werden (etwa wie bei SEQUEL). Die mit *output*-Recht versehenen Attribute der temporären Relationen könnten dann in der Host-Language weiter verarbeitet werden.

Im folgenden wird die Datenbanksprache ELSL exemplarisch vorgestellt. Sie ist in erster Linie als Vorschlag für eine Allgemeine Datenbankschnittstelle im bibliothekarischen Bereich gedacht, analog der allgemeinen Datenbankschnittstelle im kommunalen Bereich; jedoch in einer universellen Form.

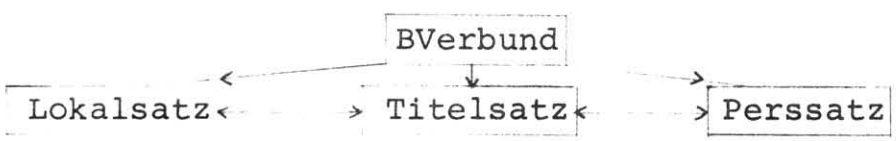
1.1 ELSL - DDL

ELSL enthält als grundlegenden Datenobjekt-Typ den Recordtyp. Ein Recordtyp dient zur Darstellung einer Klasse von Entities aus dem Informationsbereich, der einer speziellen Anwendung eines Datenbanksystems zugrundeliegt. Jedem Element (= Record, Tupel) eines Recordtyps entspricht ein bestimmtes Entity. Recordtypen sind aufgebaut wie Codd'sche Relationen, d.h. jeder Recordtyp ist unterteilt in Attribute, die einen bestimmten Wertebereich haben. Im Gegensatz zu den Codd'schen Relationen müssen Recordtypen jedoch keine eindeutigen Schlüssel haben.

Die Definition eines Recordtyps erfolgt in gewohnter Weise; z.B. für Teile der (vereinfachten) Datenbank eines Bibliotheksverbundes: [1]

---

[1] Der Beispiel-Datenbank liegt folgende Struktur zugrunde:



```
define record titelsatz
        attribute titel     char 100
        attribute verlag    char 20
        attribute ejahr     integer 4
        attribute autor_nr  integer 7
        attribute tisa_nr   integer 8

define record perssatz
        attribute vorname   char 10
        attribute nachname  char 20
        attribute autor_nr  integer 10

define record lokalsatz
        attribute tisa_nr   integer 20
        attribute standort  char 20
        attribute signatur  char 10
        attribute bestand   integer 5
```

Keines der Attribute in den obigen Recordtyp-Definitionen muß Schlüssel sein. Wird die Eindeutigkeit gewünscht, etwa beim Attribut *nachname* , so kann dies durch eine Integritätsbedingung formuliert werden.

Die Definition eines Recordtyps bewirkt einen Eintrag in die datenbankspezifischen Systemtabellen (in der Literatur auch als dictionary, directory oder metadatabase bezeichnet). Die Kreation des Recordtyps, d.h. die Bereitstellung von Hintergrundspeicherplatz, erfolgt bei der ersten schreibenden Operation auf dem Recordtyp. Der gewünschte Maximalumfang eines Recordtyps kann bei seiner Definition angegeben werden (dies bedeutet nicht, daß von Anfang an soviel Platz physikalisch belegt wird!):

```
define record titelsatz
        attribute ...
        :
        to max 10.000000 tupels
```

Ein weiterer Datenobjekt-Typ der Sprache ELSL ist der Selektor. Ein Selektor ist eine benannte Teilmenge von Records eines Recordtyps. Die Teilmenge wird durch einen booleschen Ausdruck ausgewählt, der aus durch ∧ bzw. ∨ verknüpften einfachen Bedingungen der Form

<attribute_name> <op> $[+|-]_{o}^{1}$ <attribute_name>

oder

<attribute_name> <op> <constant>

besteht, wobei <op> ∈ {> , >= , < , <= , = , !=}.

Selektoren stellen damit auf logischer Ebene einen Zugriffspfad für qualifizierte Teilmengen von Recordtypen dar. Beispiele für die Definition von Selektoren sind etwa:

define selector autorname of perssatz
where nachname = 'MUELLER' and vorname = 'HEINRICH'

Zur Definition von Selektoren können auch bereits definierte Selektoren in der Form *select* <selector_name> herangezogen werden, z.B.:

define selector autorname_nr of perssatz
where autor_nr < 500 and select autorname

Die gesamte boolesche Bedingung zur Definition eines Selektors wird im folgenden als "Selektions-Bedingung" bezeichnet.

Beziehungen zwischen zwei Recordtypen können durch sogenannte Links dargestellt werden. Links verknüpfen Records zweier nicht notwendig verschiedener Recordtypen aufgrund eines booleschen Ausdrucks mit einfachen Bedingungen der Form

$[$<record_name>.$]_{o}^{1}$ <attribut_name> <op>
$[+|-]_{o}^{1}$ $[$<record_name>.$]_{o}^{1}$ <attribute_name>

Der Recordname muß hinzugefügt werden, falls die Attributnamen nicht relativ zu den zwei beteiligten Recordtypen eindeutig sind. Zwei Beispiele sollen die Definition von Links erläutern:

```
define link titel_nachname between titelsatz and perssatz
        where verlag = nachname
```

```
define link titelsatz_perssatz between titelsatz and perssatz
        where titelsatz.autor_nr = perssatz.autor_nr
```

Links sind in beiden Richtungen definiert. Mit Links können 1:1, 1:n und m:n Beziehungen definiert werden.

Sowohl bei der Definition von Links wie bei der Definition von Selektoren werden vom System keinerlei physikalische Zugriffspfade, etwa in Form von Adreßlisten, angelegt; es erfolgt lediglich ein Eintrag der Definition in die Systemtabellen. Erst bei der Kreation werden die Zugriffspfade physikalisch eingerichtet.

Definitionen von Recordtypen, Selektoren und Links können rückgängig gemacht werden durch *undefine* ... , z.B.

```
undefine record titelsatz

undefine selector autorname

undefine link titelsatz_perssatz
```

Dabei wird die entsprechende Definitions-Information aus den Systemtabellen entfernt. Außerdem werden die jeweiligen Objekte physikalisch gelöscht (zerstört), falls sie noch kreiert sind.

1.2 ELSL - DML

Für das Einfügen, Löschen und Ändern von Informationen in Recordtypen stehen in ELSL eine Reihe von Anweisungen[1] zur Verfügung. Alle diese Anweisungen - bis auf das Einfügen eines Records - beziehen sich auf Mengen von Records und nicht auf einzelne Records. Ist bei der Ausführung einer Manipulations-Anweisung der angesprochene Recordtyp noch nicht kreiert, so wird er zu diesem Zeitpunkt kreiert.

Zunächst ist es möglich, den Inhalt einer Datei oder einer beim Retrieval erzeugten temporären Relation in einen Recordtyp zu laden:

clear and load titelsatz from file tfile

load titelsatz from relation new_titelsatz

Im ersten Fall wird der Inhalt des Recordtyps gelöscht und durch den Inhalt der Datei ersetzt. Im zweiten Fall wird der Inhalt der Relation zum Inhalt des Recordtyps hinzugefügt. Auf jeden Fall müssen Recordtyp und Datei bzw. temporäre Relation hinsichtlich ihres Aufbaus kompatibel sein.

Recordtypen können auf eine Datei gesichert werden:

save titelsatz to tfile

Diese Sicherung ermöglicht es, einen Recordtyp bei Bedarf mit *clear and load* auf einen früheren Stand zurückzusetzen.

---

[1] eigentlich: Anweisungstypen

Das Einfügen eines einzelnen neuen Records in einen Recordtyp geschieht unter Angabe von Attributnamen und -werten, z.B.

```
insert titelsatz with tisa_nr > = 95348
```

Nicht aufgezählte Attribute werden mit dem Wert "undefiniert" (geschrieben: '_') belegt.

Zum Löschen von Recordmengen ist der Recordtyp und eine Selektions-Bedingung anzugeben:

```
delete titelsatz where tisa_nr < 500
```

Beim Ändern von Recordmengen sind zusätzlich zur Selektions-Bedingung die zu ändernden Attribute mit ihren neuen Werten anzugeben:

```
update perssatz where select autorname and autor_nr = 20000
                with autor_nr = 30000
```

Das Löschen bzw. Ändern eines einzelnen Records erfolgt als Spezialfall der obigen Anweisungstypen durch die Auswahl einer einelementigen Recordmenge in der Selektions-Bedingung.

Mit einer weiteren Anweisung kann der Maximalumfang eines Recordtyps neu festgelegt werden:

```
expand titelsatz to max 15.000000 tupels
```

1.3 Kreation von Indexen, Selektoren und Links

ELSL bietet die Möglichkeit Recordtypen über ein bestimmtes Attribut bzw. eine Kombination von Attributen zu invertieren. Dieses Attribut bzw. diese Attributkombination muß nicht identifizieren (d.h. Schlüssel) sein.

create index of titel in titelsatz

create index of nachname vorname in perssatz

Mit diesen Anweisungen werden Indexe (invertierte Dateien) eingerichtet. Indexe können wieder zerstört werden:

destroy index of titel in titelsatz

ELSL kennt als sichtbare logische "Zugriffspfade" die Typen Selektor und Link. Die Definition dieser Typen wurde in Kapitel 2 erläutert. Selektoren und Links können nach ihrer Definition mit expliziten Anweisungen kreiert und wieder zerstört werden.

Bei der Kreation eines Selektors (Links) werden physikalische Verweisstrukturen zur Realisierung des Zugriffspfades eingerichtet. Ein kreierter Selektor (Link) wird bei jeder Änderung eines Recordtyps, auf dem der Selektor (Link) definiert ist, mainteniert. Soll ein Selektor (Link) auf längere Zeit unverändert bleiben, so muß er explizit gesperrt werden.

Die Kreation eines Selektors erfolgt z.B. durch

create selector autorname

Zerstört wird ein Selektor z.B. auf folgende Weise:

destroy selector autorname

Links sind zwischen zwei Recordtypen definiert; sie können jedoch, falls gewünscht, in einer Richtung kreiert bzw. zerstört werden:

create link titel_nachname from titelsatz to perssatz

destroy link titel_nachname from titelsatz to perssatz

Wird keine Richtung angegeben, so erfolgt die Kreation bzw. die Zerstörung in beiden Richtungen:

create link titelsatz_perssatz

destroy link titelsatz_perssatz

Bei der Zerstörung eines Selektors (Links) bleibt die Definition des Selektors (Links) in den Systemtabellen erhalten.

Selektoren und Links können vom Benutzer auch ohne explizite Kreation verwendet werden. In diesem Fall erfolgt eine dynamische Kreation durch das System für die Ausführung der Anweisung, in der der Selektor (Link) verwendet wird. Nach Ausführung einer Anweisung werden die dynamisch kreierten Selektoren (Links) wieder zerstört.

Zur Vereinfachung können die Anweisungen *define* und *create* für Selektoren (Links) zusammengezogen werden zu:

define and create ....

oder

defcreate ...

Das Format dieser Anweisung entspricht dem der zugehörigen *define*-Anweisung. Bei Links erfolgt in diesem Fall die Kreation in beiden Richtungen.

1.4 Sperren und Transaktionen

In ELSL muß zwischen zwei Arten von Sperren unterschieden werden: zwischen den impliziten und den expliziten Sperren.

Jede ELSL-Anweisung wird vom System als eine Verarbeitungseinheit angesehen, während der die angesprochenen Objekte implizit zum Lesen bzw. zum Schreiben - je nach Art der Anweisung - gesperrt werden. Da ELSL-Anweisungen nicht geschachtelt werden können, kann bei impliziten Sperren kein Deadlock auftreten.

Mit expliziten Sperren können vom Benutzer Datenobjekte über mehr als eine ELSL-Anweisung gesperrt werden. In ELSL können Recordtypen, Teilmengen von Recordtypen, bestimmte Attribute eines Recordtyps, Selektoren und Links gesperrt werden. Die zu sperrende Teilmenge eines Recordtyps wird durch ein Prädikat spezifiziert (Prädikatensperre). Sperren von Selektoren und Links werden auf Sperren der beteiligten Recordtypen zurückgeführt.

Eine explizite Sperre eines Recordtyps hat in ELSL folgenden allgemeinen Aufbau:

(S)
$$\underline{lock}\ z_1 A_{11} A_{12} \ldots A_{1j_1}\ z_2 A_{21} \ldots A_{2j_2} \ldots z_n A_{n1} \ldots A_{nj_n}\ \underline{in}\ \underline{record}\ R\ [\underline{where}\ P_R]_0^1$$

bzw.

$$\underline{lock}\ z\ \underline{all}\ \underline{in}\ \underline{record}\ R\ [\underline{where}\ P_R]_0^1$$

Dabei bedeutet:

- R : Recordtyp
- P_R : Prädikat auf dem Recordtyp R
- A_{ij} : Attribute des Recordtyps R
- z_i, z : Art der Sperre auf den Attributen A_{ij} bzw. auf dem Recordtyp R

In ELSL gibt es folgende Sperrarten:

read : Lesesperre (für konsistentes Lesen)
write : Schreibsperre
readiw : Lesesperre mit beabsichtigter Schreibsperre (verhindert, daß zwischen Lesephase und Schreibphase ein anderer Benutzer auf den betreffenden Datenobjekten schreiben kann)

Die Verträglichkeit der Sperrarten ist wie folgt definiert:

	read	*readiw*	*write*
read	+	+	-
readiw	+	-	-
write	-	-	-

+ : verträglich
- : nicht verträglich

Als Prädikate sind in ELSL Selektions-Bedingungen

erlaubt, wobei jeder Ausdruck *select s* vom System durch die Bedingung des Selektors s substituiert wird.

Ausgehend von (S) können die Sperren auf Recordtypen in 4 Klassen unterteilt werden:

(S1) Sperre eines gesamten Recordtyps, z.B.

```
lock read all in record perssatz
```

(S2) Sperre von Attributen eines Recordtyps, z.B.

```
lock read vorname nachname write autor_nr ...
     in record perssatz
```

(S3) Sperre einer Teilmenge eines Recordtyps, z.B.

```
lock write all in record perssatz
     where nachname = 'MUELLER'
```

(S4) Sperren von Attributen einer Teilmenge eines Recordtyps, z.B.

```
lock read nachname readiw autor_nr
     in record perssatz, where autor_nr < 588
```

Sind auf explizit mit *readiw* oder *write* zu sperrenden (Teilen von) Recordtypen Integritätsbedingungen definiert und aktiviert (vgl. Kap. 4.1) so werden vom System automatisch alle für die Integritätsprüfung zusätzlich notwendigen Datenobjekte mit *read*-Sperre versehen. Beim Sperren von Attributen (mit *readiw* oder *write*), die in Link-Bedingungen von kreierten Links vorkommen, wird für diese Links ebenfalls automatisch eine *read*-Sperre gesetzt.

Das Sperren von Selektoren, z.B.

lock read all in selector autorname

wird intern zurückgeführt auf eine Sperre des Typs (S2) bzw. (S4).

Ähnlich wird beim Sperren von Links, z.B.

lock read link titel_nachname

die Sperre intern ersetzt durch Sperren des Typs (S2) für die in der Link-Bedingung enthaltenen Attribute der beteiligten Recordtypen.

Um Deadlocks zu verhindern und ein konsistentes Rücksetzen bei Integritätsverletzungen zu ermöglichen, unterliegt das explizite Sperren in ELSL folgenden Einschränkungen:

- Am Anfang einer Sperrphase müssen alle Sperren auf einmal gegeben werden (Ausnahme *readiw* s. unten); d.h. die Sperr-Anweisungen dürfen von keiner anderen ELSL-Anweisung unterbrochen werden.

- Nachforderung von Sperren innerhalb der Sperrphase ist nur für Objekte, die mit *readiw* gesperrt wurden, möglich.

- Alle Sperren müssen auf einmal freigegeben werden. Das geschieht durch die Anweisung

  unlock

- Erst nach Beendigung einer Sperrphase durch *unlock* dürfen wieder neue Sperren gesetzt werden.

- Die impliziten Sperren innerhalb einer Sperrphase müssen von den expliziten Sperren überdeckt werden. Ist dies nicht der Fall, so wird mit Fehlermeldung auf den Datenbankzustand am Anfang der Sperrphase zurückgesetzt[1].

Damit kommt in ELSL eine Verschärfung des konsistenten Lockprotokolls zur Anwendung. Selbstverständlich können von einem Benutzer nur die Datenobjekte gesperrt werden, für die er entsprechende Zugriffsrechte besitzt.

Eine Sperrphase in ELSL stellt eine Transaktion unter expliziter Angabe der Sperren dar. Deshalb sind in ELSL keine anderen Sprachmittel, die den Beginn, bzw. das Ende einer Transaktion anzeigen, notwendig.

---

[1] Es besteht jedoch auch die Möglichkeit, daß vom System in diesem Fall automatisch nachgesperrt wird, falls die nachträglichen Sperren nicht in Konflikt mit Sperren anderer Benutzer stehen.

1.5 Kontrollfunktionen

Integritätsbedingungen

Zur Überprüfung und Erhaltung der semantischen Integrität der Daten können in ELSL Integritätsbedingungen (integrity constraints) formuliert werden. Diese Integritätsbedingungen beschreiben einen korrekten Zustand der zugrundegelegten Datenbank - d.h. erlaubte Werte von Attributen, Beziehungen zwischen Attributen etc. - und zulässige Übergänge von einem Zustand der Datenbank in einen anderen.

Eine Integritätsbedingung in ELSL setzt sich zusammen aus:

(1) dem Namen der Integritätsbedingung

(2) den Namen der Recordtypen, auf denen die Integritätsbedingung definiert ist

(3) der eigentlichen Bedingung (*condition*), die
   a) auf den Recordtypen erfüllt sein muß, bzw.
   b) korrekte Änderungen innerhalb von Recordtypen beschreibt

(4) der Auslöseregel (*event*), die angibt,
   a) bei welchen ELSL-Anweisungen die Bedingung zu prüfen ist und
      ob die Bedingung unmittelbar nach der entsprechenden Anweisung (*immediate*) oder am Ende einer Sperrphase (*deferred*) zu prüfen ist
   b) oder ob die Bedingung durch einen expliziten Aufruf (*by check*) überprüft werden soll.

(5) der Reaktionsregel (*reaction*), die angibt, wie das System bei einer Verletzung der Bedingung reagiert.

Vier Möglichkeiten werden hier angeboten:

reset : Rücksetzen auf den Zustand zu Beginn der aktuellen Sperrphase und Meldung

report : Meldung und Ausführung der Anweisung

noexec : Anweisung wird nicht ausgeführt, d.h. übergangen, und es wird eine Meldung ausgegeben

noincons : Daten, die die Integritätsbedingung verletzen, werden nicht eingetragen. Außerdem erfolgt eine Meldung.

Alle von Date geforderten Arten von "integrity constraints" können in ELSL als eigentliche Bedingung (*condition*) formuliert werden.

In ELSL gibt es fünf verschiedene Anweisungstypen für Integritätsbedingungen:

(I1) Definition (*define iconstraint*):

Die Definition bewirkt einen Eintrag in die Systemtabellen, die Integritätsbedingung wird jedoch noch nicht überprüft.

(I2) Aktivierung (*activate*):

Vom Zeitpunkt der Aktivierung an wird die Integritätsbedingung bei den in der Auslöseregel angegebenen Ereignissen überprüft.

activate c1

activate c3

activate c2 without backcheck

Bei der Aktivierung werden die bereits vorhandenen Daten daraufhin überprüft, ob sie die Bedingung (*condition*) erfüllen. Diese Überprüfung kann durch *without backcheck* außer Kraft gesetzt werden.

(I3) Inaktivierung (*inactivate*):

Die Aktivierung einer Integritätsbedingung kann wieder rückgängig gemacht werden, d.h. die Überprüfung wird außer Kraft gesetzt.

inactivate *c1*

(I4) Aufheben der Definition (*undefine iconstraint*) :

Mit der Anweisung *undefine iconstraint* wird die Definition einer Integritätsbedingung aus den Systemtabellen entfernt. Damit ist die Integritätsbedingung dem System nicht mehr bekannt.

undefine iconstraint *c1*

(I5) Explizite Überprüfung (*check*):

Eine explizite Überprüfung ist nur möglich für Integritätsbedingungen mit *by check* als Ereignis in der Auslöseregel.

check *c3*

Ein vollständiger Überblick über die in ELSL möglichen Integritätsbedingungen wird in [ASCHENBRENNER 78] gegeben.

1.6 Autorisierung

ELSL enthält Anweisungen, die die Vergabe von Zugriffsrechten auf Datenobjekte ermöglichen. Vorausgesetzt wird, daß sich jeder Benutzer beim Eröffnen einer Datenbank durch ein spezielles Benutzerkennzeichen zu identifizieren hat - evtl. verbunden mit einem Identifizierungsmechanismus.

Bei der Realisierung der Zugriffskontrolle in ELSL wurde von folgenden Voraussetzungen ausgegangen:

(1) Zugriffsrechte können vergeben werden auf Recordtypen, Teilmengen und Attributen von Recordtypen, Selektoren, Links und Integritätsbedingungen.

(2) Zugriffsrechte erlauben einem Benutzer bestimmte ELSL-Anweisungen auf bestimmten Objekten auszuführen.

(3) Die Zugriffsrechte werden zentral durch den Datenbankadministrator (besitzt privilegiertes Kennzeichen) vergeben. Eine Weitergabe von Zugriffsrechten ist nicht möglich.

Die Zugriffskontrolle wird also aus logischer Sicht durch eine Berechtigungsmatrix realisiert, die angibt, welche Anweisungen ein Benutzer auf welchen Datenobjekten ausführen darf.

Im folgenden sollen die ELSL-Anweisungen für die Vergabe von Zugriffsrechten an einigen Beispielen erläutert werden. Eine Liste aller Zugriffsrechte mit ihrer Bedeutung ist im Anhang in den Anmerkungen zur ELSL-Syntax zu finden.

(a) Benutzer 001 erhält alle Rechte auf dem Recordtyp *customers:*

give all rights on record titelsatz to 001;

(b) Die Benutzer 002 und 003 dürfen den titel aller titelsätze lesen (d.h. daraus temporäre Relationen bilden):

give read on attribute titel in titelsatz to 002 003

(c) Benutzer 004 darf das Attribut *nachname* im Recordtyp *perssatz* lesen und ändern:

give read update on attribute nachname in perssatz to 004

(d) Dem Benutzer 004 wird das Recht, den Nachnamen zu ändern, entzogen:

remove update on attribute nachname in perssatz from 004

(e) Benutzer 005 darf Integritätsbedingungen auf dem Recordtyp *credits* definieren (und damit auch aktivieren etc.):

give define iconstraint on record perssatz to 005

(f) Benutzer 007 darf Recordtypen definieren bis zu einem bestimmten Maximalumfang; er erhält damit auch alle Rechte auf den von ihm definierten Recordtypen:

give define record to 007 for max 5 k bytes

1.7 Retrieval

In ELSL wird wie in LSL unterschieden zwischen permanenten Recordtypen (*record*), die in der Datenbank abgespeichert sind, und temporären Recordtypen (*relation*), die für das Retrieval von Informationen aus der Datenbank gebildet werden. Im folgenden werden die temporären Recordtypen als (temporäre) Relationen bezeichnet.
Für das Retrieval stehen alle Möglichkeiten der Relationen-Algebra zur Verfügung. Das Retrieval kann in mehreren Schritten erfolgen. Ergebnis eines Retrieval-Schrittes ist immer eine temporäre Relation, die entweder

(R) aus mehreren Recordtypen bzw. Relationen durch Anwendung von Selektions-Bedingungen ($\hat{=}$ Restriktion), Link-Bedingungen ($\hat{=}$ allgemeiner Join) und Projektionen

oder

(R') aus zwei Recordtypen bzw. Relationen gleichen Typs (d.h. domain-kompatibel) durch Anwendung der Operationen *union* (Vereinigung), *intersection* (Durchschnitt) und *difference* (Differenz)

gebildet werden.

Relationen können wiederum definiert (*define relation*), kreiert (*create relation*), zerstört (*destroy relation*) werden, und ihre Definition kann rückgängig gemacht werden (*undefine relation*). Definition und Kreation können zusammengezogen werden (*define and create relation*).

Auf Relationen können jedoch keine Selektoren, Links oder Integritätsbedingungen definiert werden.

Die folgenden Beispiele sollen einen Überblick über die Definition von Relationen geben. Dabei ist zu beachten, daß ein Benutzer zur Bildung von Relationen nur diejenigen Datenobjekte heranziehen kann, für die er ein *read*-Recht besitzt.

Recordtypen oder Relationen können auf eine Datei oder auf das Normalausgabemedium ausgegeben werden. Dabei können Selektions-Bedingungen und Projektionen angegeben werden. Auf allen auszugebenden Attributen muß *output*-Recht bestehen.

Es besteht in ELSL außerdem die Möglichkeit, statistische Funktionen für numerische Attribute zu berechnen und auszugeben. Dabei kann man sich wiederum auch auf Teilmengen von Recordtypen beziehen.

In ELSL kann demnach bei der Vergabe von Rechten unterschieden werden zwischen dem Recht, Werte eines Attributs auszugeben (*output*-Recht), und dem Recht, Relationen zu bilden und Funktionen von Attributwerten zu ermitteln, ohne die Attributwerte selbst sehen zu dürfen (*read*-Recht).

Temporäre Relationen können in (permanente) Recordtypen umgewandelt werden:

1.8 Technische Funktionen

Zur Bearbeitung einer Datenbank sind eine Reihe von technischen Funktionen, wie z.B. das Eröffnen oder das Schließen einer Datenbank, notwendig. ELSL wurde so konzipiert, daß gleichzeitig mehrere Datenbanken bearbeitet werden können. Zu jedem Zeitpunkt einer ELSL-Bearbeitung gibt es genau eine aktuelle Datenbank - vorausgesetzt, daß mindestens eine Datenbank geöffnet ist -. Alle Namen von Records, Selektoren etc., die nicht durch einen Datenbanknamen qualifiziert sind, beziehen sich auf die aktuelle Datenbank.

Jede Datenbank muß vor ihrer Bearbeitung eröffnet werden. Dabei ist ein Benutzerkennzeichen anzugeben, z.B.

open db1 for 001

Anschließend kann noch ein spezielles Identifizierungsverfahren ablaufen, wie es etwa in [ZZ 74] beschrieben ist.

Die eröffnete Datenbank wird zur aktuellen Datenbank.

Ein Wechsel der aktuellen Datenbank wird erreicht durch die Eröffnung einer neuen Datenbank oder explizit durch:

switch to <database_name>

Nach ihrer Bearbeitung muß eine Datenbank abgeschlossen werden:

close db1

Schließlich kann eine Datenbank auf den letzten Recoverypunkt (das ist in ELSL der Beginn der aktuellen Sperrphase) zurückgesetzt werden

reset db1

1.9 Informationsfunktionen

In den derzeit verbreiteten Datenbanksprachen sind nur unzulängliche Möglichkeiten vorgesehen, den strukturellen Aufbau und Inhalt einer Datenbank abzufragen. In ELSL wurden zu diesem Zweck sogenannte "Informationsfunktionen" eingeführt. Diese Funktionen geben in bestimmtem Format auf dem Normalausgabemedium Informationen über den Gesamtaufbau einer Datenbank, über den Aufbau von Recordtypen etc. aus. Dabei werden nur die Informationen ausgegeben, die der anfragende Benutzer sehen darf, d.h. abhängig von den Zugriffsrechten des Benutzers.

Beispiele:

(a) Es soll Information über den Gesamtaufbau der aktuellen Datenbank ausgegeben werden:

inform about database

(b) Es soll über den Inhalt der Integritätsbedingung c1 informiert werden:

inform about iconstraint c1

(c) Der DBA will sich über alle Zugriffsrechte des Benutzers 001 informieren:

inform about authorizations for 001

(d) Die Definitionen aller Recordtypen sollen aufgelistet werden:

inform about all records

2. Beschreibung der Komponente DFÜR

Es wird zunächst nur die unter I/2.1 genannte Teilkomponente beschrieben und diese nur in Kurzform. Eine ausführliche Beschreibung wird in [FRY 77] gegeben. Auf die unter I/2.2 bzw. I/2.3 wird in dieser Arbeit nicht näher eingegangen. Es ist lediglich anzumerken, daß als TDL die Sprache CONVERT in Fräge käme.

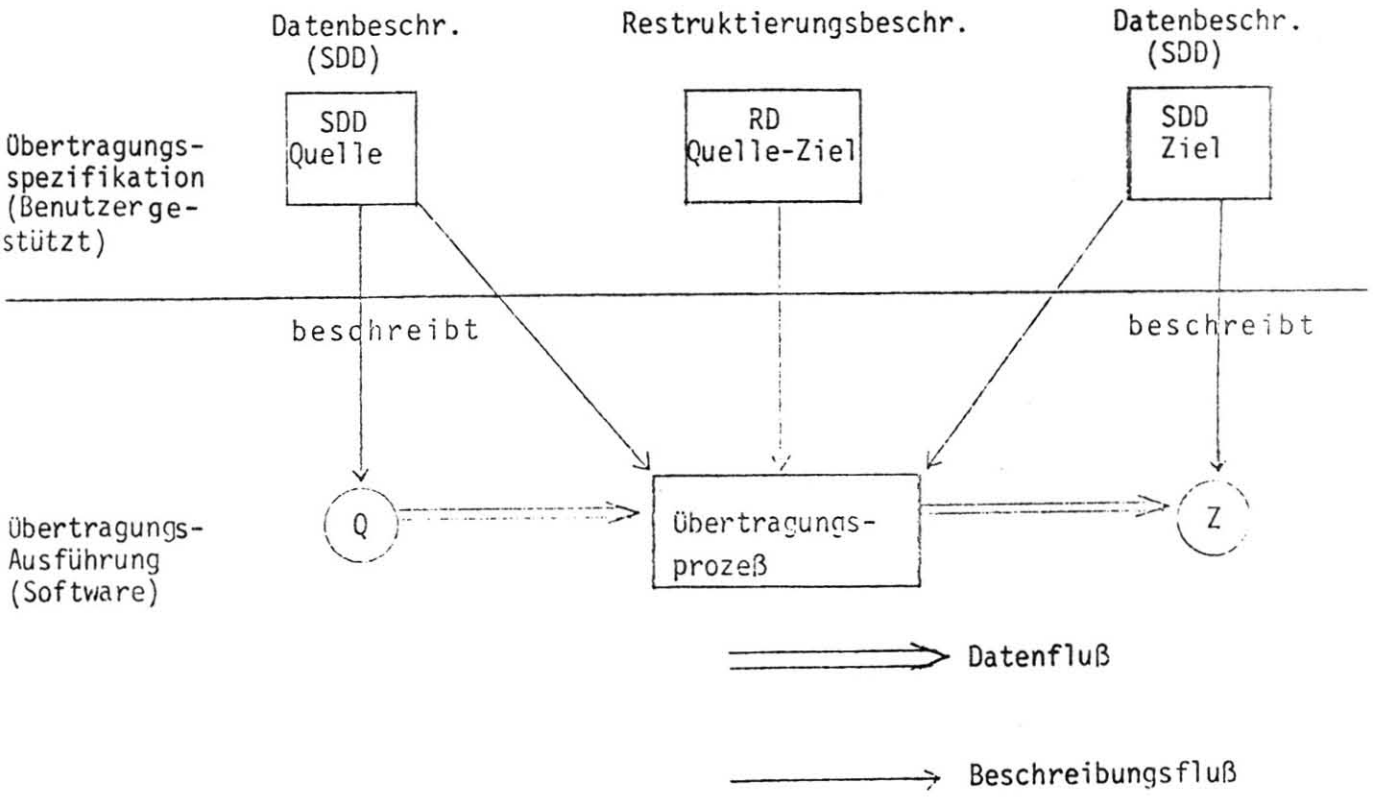
2.1 Beschreibung der SDDL

Zweck einer SDDL ist es bereits gespeicherte und zukünftig zu speichernde Daten detailliert zu beschreiben, sodaß sie von einem System in das andere transferiert werden können.

Die Übertragung verläuft in zwei Phasen:

1) Vorbereitung der Übertragungsspezifikationen

2) Ausführen der Übertragungsspezifikationen



Die Ausführung der Übertragungsspezifikation besteht aus drei verschiedenen Programm-Moduln:

1) Zugriff auf die Quelldaten- lesend

2) Durchführen der notwendigen Transformationen auf einer internen Darstellung der Quelle-Restrukturierung und

3) Erzeugen der Zieldaten- schreibend.

Eingabe für den ersten Prozeß sind drei vom Benutzer gelieferte Beschreibungen:

(i) Logische Zugriffsstruktur (LAS)
(ii) Physikalische Zugriffsstruktur (PAS)
(III) Eine Lese-Spezifikation (RS) für das Durchlaufen von (i), (ii).

} Komponenten der SDDL

Die Aufgabe der zweiten Funktion im Leseprozeß ist die Erzeugung einer selbstschreibenden Darstellung der Quelldaten, die Übersetzer-Interne-Form (TIF).

Der zweite Prozeß in der Übertragungsausführung ist die Restrukturierung. Die Aufgabe dieses Programm-Moduls ist die Durchführung der logischen Transformation auf den Quelldaten, die von der Restrukturierungsbeschreibung (RD) gefordert werden.

Eingabe für den Restrukt-rierungsprozeß sind:

(i) die LAS der Quelldaten
(ii) die LAS der Zieldaten
(III) die TIF der Quelldaten und
(iv) die Restrukturierungsbeschreibung.

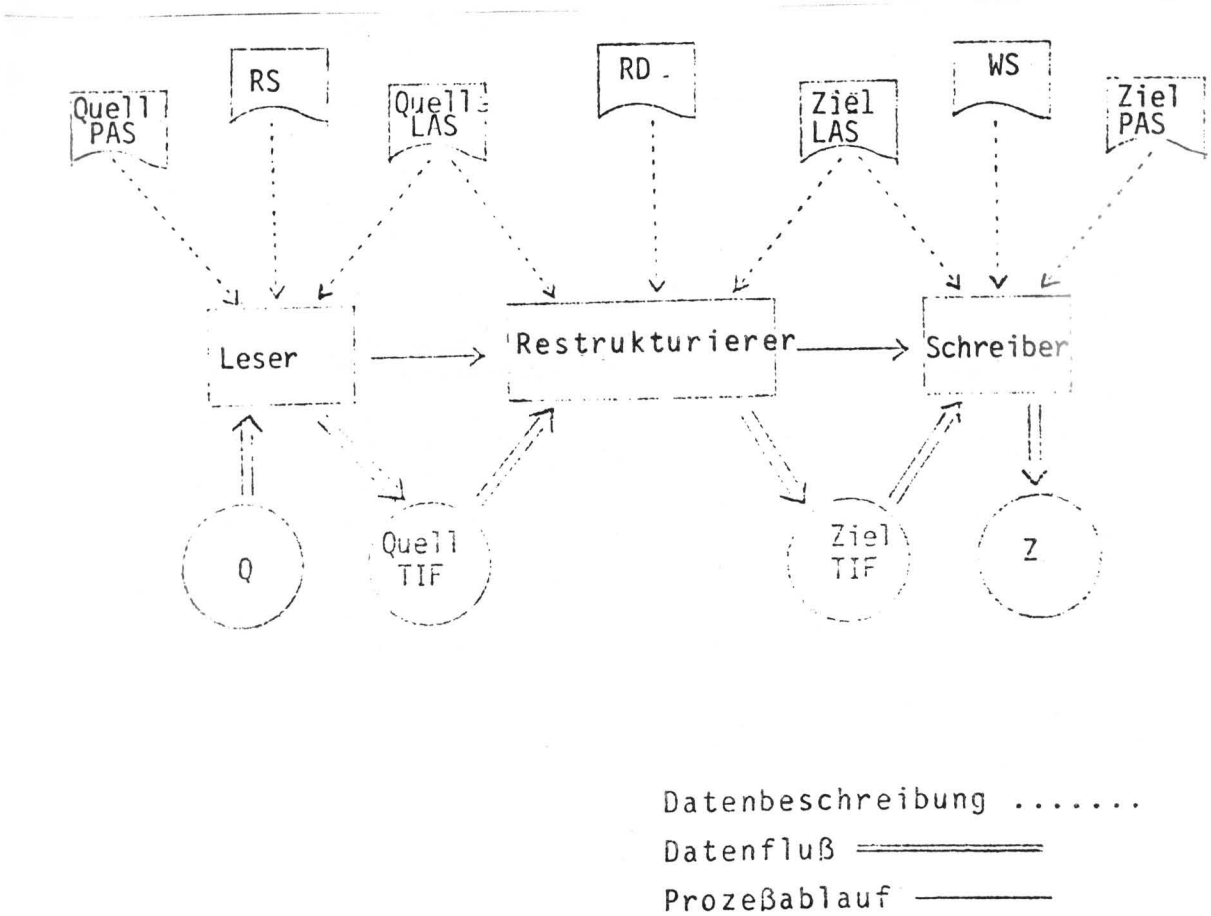
Unter der Kontrolle der RD führt der Restrukturierer die notwendigen Transformationen aus und erzeugt die Zielversion der Daten in TIF.

Als invers zum Lesevorgang kann der Schreibvorgang angesehen werden, In erster Linie werden beim Schreibvorgang die Zieldaten erzeugt, entsprechend den Spezifikationen der SDDL.

Eingabe für den Schreibvorgang sind die Zielversionen von

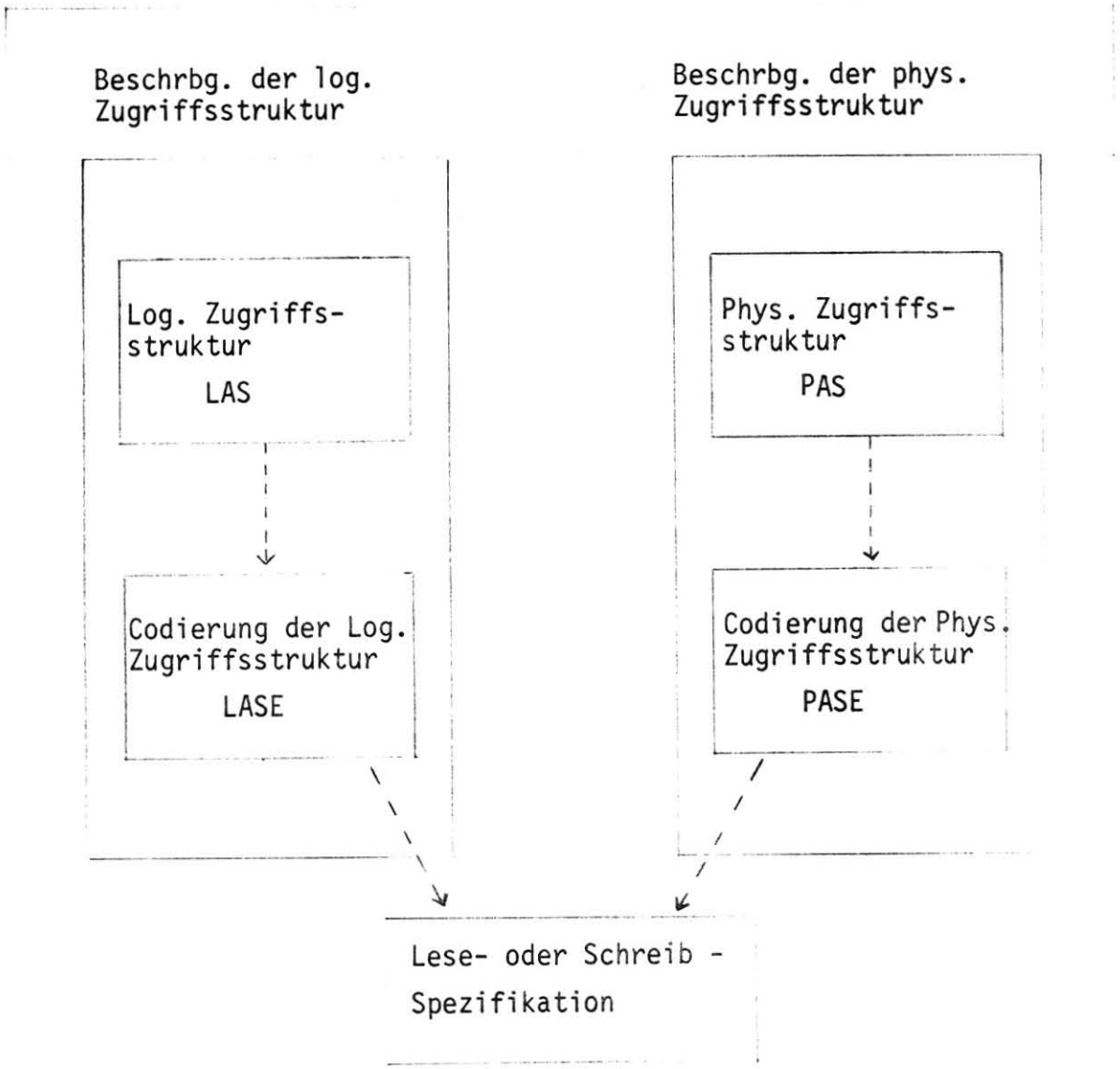
(i) TIF
(ii) LAS
(iii) PAS
(iv) Schreibspezifikation (WS).

In der folgenden Abbildung ist der Übertragungsvorgang schematisch dargestellt:



Die folgende Abbildung zeigt schematisch die Beziehungen zwischen den Komponenten einer SDDL - Beschreibung:

SDDL-Beschreibung von Quell- oder Zieldaten



Wie in der Beschreibung der ELSL gezeigt erfüllt diese Sprache alle Forderungen, die sich aus den bibliothekarischen Daten und ihre Verarbeitung unter Verbundaspekten ergeben. Lediglich die Forderung nach variabler Satz- und Feldlänge ist nicht erfüllt. Dieser Bedingung genügt aber kein bisher bekanntes Datenverwaltungssystem. Sie läßt sich aber gegebenenfalls auf Umwegen erfüllen (analog DBS440/TELDOK).

Was nun die SDDL angeht, so verlangt diese Sprache eine Beschreibung der Zugriffspfade auf logischer Ebene (LAS). Dies ist bei CODASYL-Systemen trivialerweise möglich, bei Relationensystemen jedoch nicht so ohne weiteres. Hier bietet sich wieder ELSL an, das auch die Möglichkeit der Beschreibung von Zugriffspfaden auf logischer Ebene bei relationalen Datenbanksystemen vorsieht. Dies garantiert die Möglichkeit der Übertragung CODASYL - RELATIONAL und RELATIONAL - RELATIONAL.

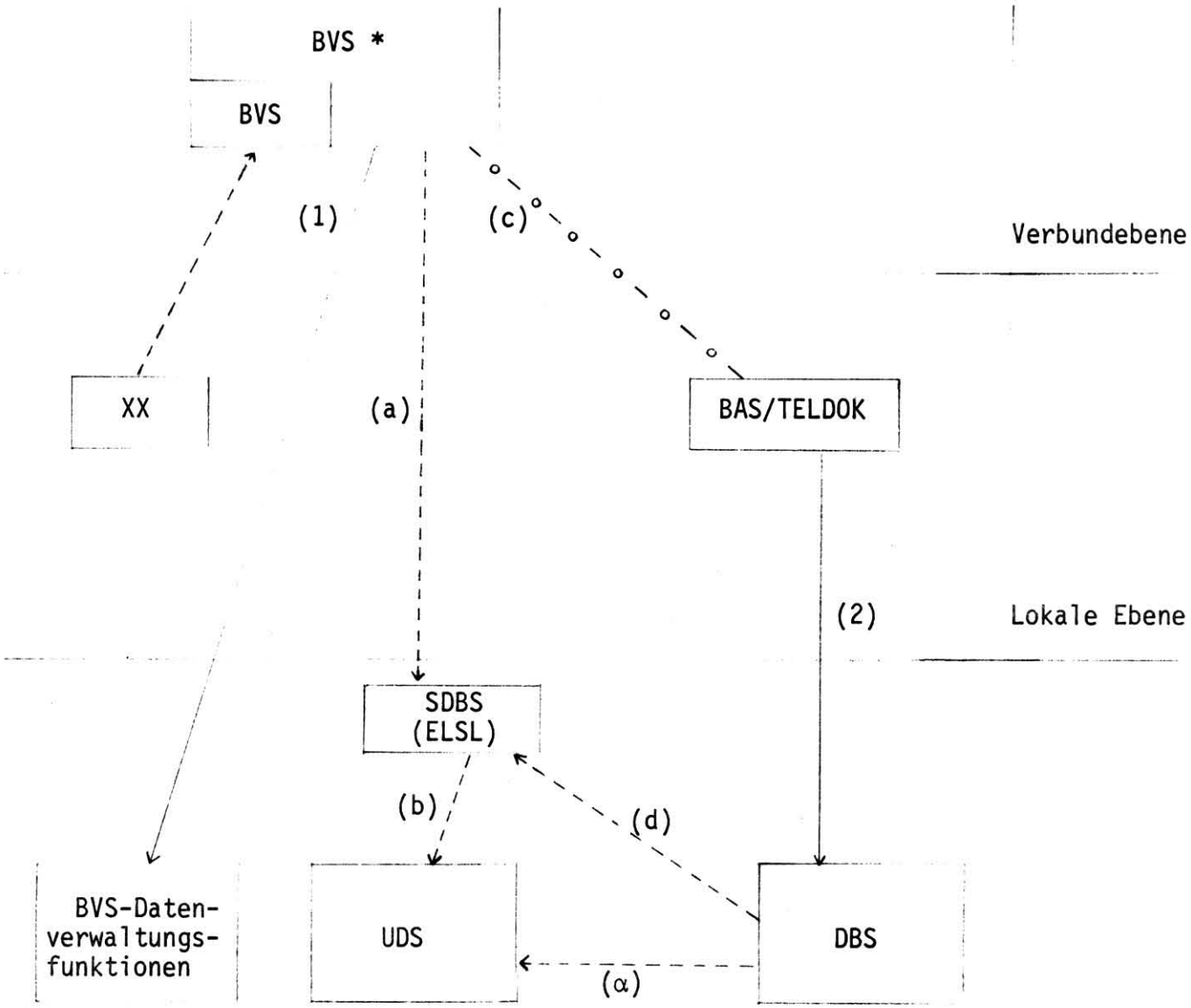
III. Integration bereits existierender Software:

Geht man von dem hier entwickelten Konzept aus und nimmt an, daß die zukünftige Software im Bibliotheksbereich auf einer gemeinsamen Schnittstelle entwickelt wird, unabhängig davon ob diese in der komfortablen in ELSL beschriebenen Form vorhanden ist oder in der minimalen Form wie im Bereich der kommunalen Datenverarbeitung, so stellt sich die Frage, wie bereits existierende Software im Bibliotheksbereich, die ja schon einige Bearbeitungsvorgänge automatisiert, in das Konzept integriert werden soll. Eine Neuentwicklung kann aus Kostengründen wohl von vorneherein ausgeschlossen werden.

An bereits existierender Bibliothekssoftware sind vor allem zu nennen: BVS (Bibliotheksverbundsystem), BIKAS II (Bibliothekskatalogisierungssystem), (KO)BAS/TELDOK ((Konstanzer)-Bibliotheksautomatisierungssystem/Telefunken-Dokumentationssystem) und DOBIS (Dortmunder Bibliothekssystem). In den folgenden Überlegungen werden in erster Linie BVS und (KO)BAS/TELDOK berücksichtigt.

In der nachfolgenden Skizze wird schematisch die Integration der Einzelsysteme in das Gesamtkonzept gezeigt.

Integrationsschema



Zeichenerklärung:

A ——→ B	benutzt
A –o–o→ B	A wird durch Erweiterung des Leistungsverhalten von B abgedeckt
A – – – → B	Zu realisierende Komponenten

XX: weitere vorhandene Bibliothekssysteme

Da BVS in seiner derzeitigen Form den in I. geforderten Leistungsumfang nicht hat, müßte das System zu BVS * erweitert werden und gleichzeitig über SDBS auf das Datenverwaltungssystem UDS abgestützt werden. Die Erweiterung müßte so vorgenommen werden, daß BVS * auch die bisherigen Leistungen von BAS/TELDOK erbringen kann (c) und daß (a), (b) zusammen (1) ersetzen. Über (d) und (b) ist die Migration von DBS-Funktionen möglich, wie bereits bei der Beschreibung von ELSL erwähnt können sowohl UDS- wie auch DBS-Funktionen in dieser Sprache beschrieben werden. Eine detaillierte Beschreibung wird Gegenstand eines weiteren Berichtes sein. Über (α) wäre eine Datenbankmigration mit Unterstützung durch Migrationsanleitung bzw. über die Komponente DFÜR möglich. Auf die Migration des TR440 auf ein Nachfolgesystem wird im nächsten Abschnitt eingegangen.

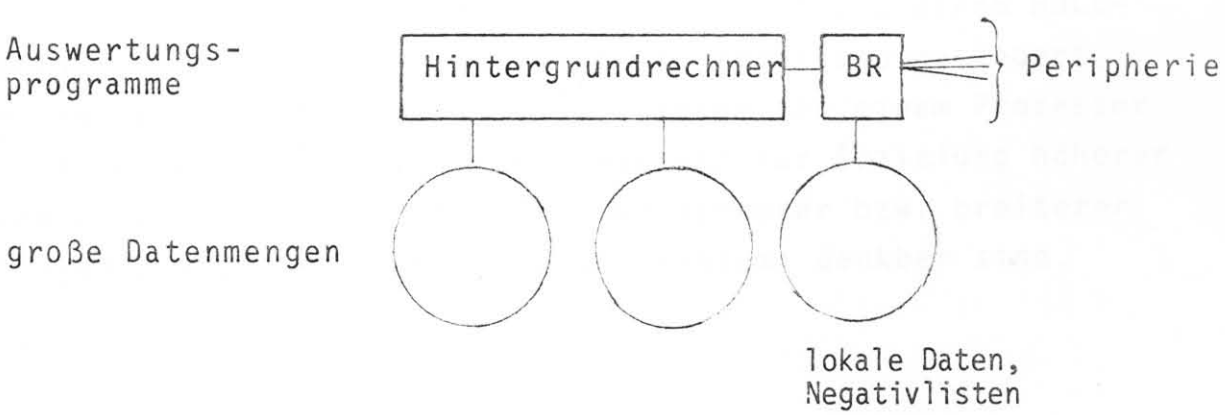
2. Migration

Das Problem der Migration des TR440 auf ein Nachfolgesystem wird hier nicht in voller Allgemeinheit behandelt, sondern nur soweit es in diesem Kontext von Bedeutung ist. Es geht hier also um die Migration der Datenhaltungsfunktionen, insbesondere des Systems DBS, und der aufsetzenden Anwenderprogramme, insbesondere des Systems BAS (und TELDOK).Für Bibliotheken ist ein nahtloser Übergang von BAS auf ein äquivalentes Programmsystem (bei Wechsel der Hardware) unabdingbar, da der Geschäftsgang entweder bereits auf BAS (oder ähnlichen Systemen) aufbaut oder die Bibliotheken im Zuge der anlaufenden Automatisierung den Einsatz von BAS gerade vorbereiten. Wir gehen für die weitere Diskussion von folgenden Voraussetzungen aus:

1. Die Abwicklung des Geschäftsganges einer Bibliothek durch EDV wird einem Bibliotheksrechner BR übertragen. BR kann darüberhinaus noch andere Funktionen erfüllen, von BR müssen jedoch gewisse Forderungen bezüglich Responszeiten, Verfügbarkeit, Datenschutz etc. garantiert werden, die üblicherweise von einem allgemeinen Dienstleistungsrechner nicht zu erfüllen sind.

2. BR soll autonom oder halbautonom, d.h. mit einem Großrechner als Hintergrundrechner gefahren werden können (vgl. A 78-18,I).

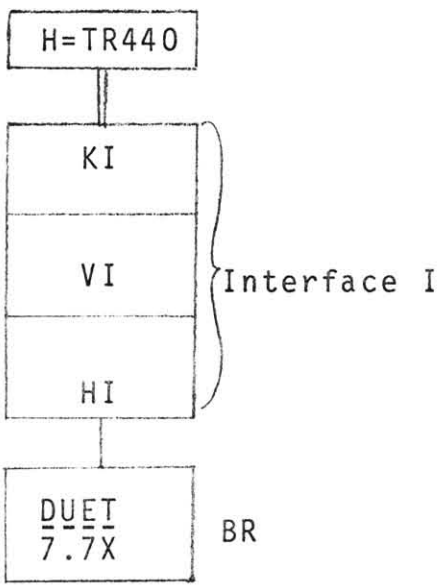
3. BR soll (aufwärts)-kompatibel zu einem TR440-Nachfolgesystem sein. Setzen wir halbautonomen Betrieb voraus, dann ergibt sich folgende Konfiguration:



Setzen wir "Hintergrundrechner = TR440", "BR = kleineres Siemenssystem" voraus, dann trägt dieser halbautonome Betrieb der jetzigen Situation am besten Rechnung und erlaubt eine schrittweise Verlagerung von Auswertungsprogrammen und Datenmengen (die im Zuge einer allgemeinen Migration zu betrachten ist) auf den BR und schließlich dessen autonomen Betrieb (ggf. als Bestandteil eines Bibliotheks-Verbundsystems). Als Voraussetzung für den halbautonomen Betrieb des BR ist also die Kopplung Hintergrundrechner (TR440) - BR (Siemens 7.7X,X=08 oder 38) zu realisieren.

2.1 Kopplung Hintergrundrechner (TR440) mit Bibliotheksrechner

Zur Kopplung von H mit BR wird folgendes Interface I entwickelt:



I besteht aus 3 (logischen) Bestandteilen: einem Kanalanpass-interface KI, einem Vermittlungsinterface VI und einem HDLC-Interface HI. Alle Interfaces sind mikroprozessorgesteuert, wobei im einfachsten Fall alle Funktionen von einem Prozessor (INTEL8080) wahrgenommen werden, während zur Erzielung höherer Geschwindigkeiten (z. B. zum Treiben mehrerer bzw. breiterer Leitungen zum BR) auch Mehrprozessorsysteme denkbar sind.

Das Kanalinterface KI nimmt die Seriell-Parallel/Parallel-Seriell-Wandlung der Bipol-Steuer-und Informationssignale des Standardkanals vor und übergibt vom TR440 ankommende Daten an das Vermittlungsinterface VI.VI führt alle nötigen Umformatierungs-(z.B. Blockungs- bzw. Entblockungs-) und Umcodierungsoperationen durch und übergibt die Daten an das HDLC-Interface HI, das diese dann (im dort verlangten Format) an den BR schickt.

Halbautonomer Betrieb des BR am TR440 heißt nun, daß der BR als Satellit zum TR440 gefahren wird und zwar in folgender Weise:

Der Benutzer (des BR) arbeitet mit den gewohnten TR440-Kommandos, die vom BR (d.h. dort existierendem Entschlüssler) automatisch an den TR440 weitergereicht und dort ausgeführt werden. Darüberhinaus kann der BR auch als eigenständiger Rechner betrieben werden, und schließlich sorgen Dateitransferkommandos XPUT (Quelldatei, Zieldatei) bzw. XGET (Quelldatei, Zieldatei) für die Übertragung von Dateien von BR$\longrightarrow$H bzw. von H$\longrightarrow$BR. Dynamisches Umschalten zwischen den beiden Betriebsarten ermöglicht lokales Arbeiten am BR (Aufbau von Dateien) und schließlich die Weiterverarbeitung am TR440.

Bemerkung: Das Interface VI ist so angelegt, daß an ihm (zumindest indirekt) auch Peripheriegeräte betrieben werden können. In diesem Fall kann der Benutzer wählen (bzw. umschalten) zwischen dem TR440 und dem Siemensrechner als ausführendem Rechner, H und BR werden hier also gleichrangig behandelt.

Für KI wird auf den Vortrag von Herrn Dipl.-Phys. B. Kett auf der TR440-Benutzertagung, September 1978, Konstanz verwiesen.

VI ist analog aufgebaut, wie der zentrale Satellitenrechner AEG80/20 zum TR440, der am Rechenzentrum des Saarlandes seit 1976 betrieben wird.

Für HI siehe: Hegger, R.: Ein intelligentes Interface für HDLC, Diplomarbeit, am FB 10 der Universität des Saarlandes, in Vorbereitung.

Literatur

[ASCHENBRENNER 78]
M. Aschenbrenner, M. Hein, H. Langendörfer: ELSL - Eine dialogorientierte Datenbanksprache für die konzeptuelle Ebene eines Datenbanksystems. Technische Universität München, Institut für Informatik, TUM-INFO-7825.

[FRY 77]
The Stored-Data Definition and Translation Task Group of the CODASYL Systems Committee. Stored-Data Description and Data Translation: A Modell and Language. Information Systems, Heft 2, März 1977.

[KOHL 78]
Bibliothekarische Anforderungen an On-line Verbundsysteme: überregional anerkannte Bearbeitungsregeln, DIN-Normen und gesetzliche Vorschriften. Ein Diskussionsbeitrag. Erscheint in: "Zeitschrift für Bibliothekswesen und Bibliographie (ZfBB)".

[LANGENDÖRFER 78a]
H. Langendörfer, H. Scheidig: Struktur eines integrierten Bibliotheks- und IuD-Informationssystems. Bericht Nr. A 78 - 18, Universität des Saarlandes, Fachbereich Angewandte Mathematik und Informatik.

[LANGENDÖRFER 78b]
H. Langendörfer, H. Martin: Konzept eines integrierten Literaturnachweis und -vermittlungssystems. Nachrichten für Dokumentation, Heft 6, 1978.