ADSS – ein erweitertes Data Dictionary

Im Jahr 1987 entwickelt und programmiert von Hans Ulrich Stalder

Abstrakt

ADSS steht für „All Direction Search System“. Im Gegensatz zu einer Datenbank, in der nur eigentliche Daten gespeichert sind, enthält ein Data Dictionary Informationen über diese Daten (sogenannte Metadaten). Mit ADSS werden die miteinander in Verbindung stehenden Ausgangsdaten in einer Kontrolldatenbank gegenseitig referenziert und in Clustern gespeichert. Ein Cluster definiert im vorliegenden Fall die Zusammengehörigkeit voneinander abhängiger Elemente oder Objekte an einem Ort oder in einem bestimmten Kontext zu einer bestimmten Zeit.

Entstehung und Untergang von ADSS

Die immer grösser werdenden Datenmengen gegenseitig abhängiger Informationen erschwerten deren Verwaltung sowie die Synchronisation. In der Folge dachte ich über ein System nach, das die Beziehungen zwischen den Daten aufzeigen kann. Grundsätzlich handelt es sich dabei um ein Data Dictionary im herkömmlichen Sinn. Zusätzlich sollte jedoch auch der zeitliche Aspekt berücksichtigt werden, nämlich wann und welche Daten verändert wurden. Das heisst, es sollte jederzeit feststellbar sein, welche Daten zu welchem Zeitpunkt miteinander in Beziehung stehen, beziehungsweise standen.

Konzeptionell werden alle Daten, für die eine Beziehung definiert wird, mit einem virtuellen Netz überzogen. An jedem Knotenpunkt konnte man, bildlich gesprochen, „ziehen“ und damit alle zugehörigen Daten – auch zu einem bestimmten Zeitpunkt in der Vergangenheit – heranziehen beziehungsweise darstellen. Dies ermöglicht es, bei Inkonsistenzen rückwirkend einen synchronen Datenzustand zu rekonstruieren.

Zu jener Zeit waren relationale Datenbanken (RDBs) noch weitgehend theoretischer Natur. Selbst Grossrechnersysteme waren damals bei komplexen Aufgaben mit RDBs überfordert. Daher entwickelte ich – neben meiner beruflichen Tätigkeit, das heisst an Wochenenden und in den Ferien – das hier beschriebene „All Direction Search System“. Mein Arbeitgeber gestattete mir für Testzwecke die Benutzung eines wassergekühlten IBM-System/360-Grossrechners, allerdings nur während der Nachtstunden.

Die auferlegte Verwendung der Programmiersprache PL/I (Programming Language One) erwies sich als anspruchsvoll. Damals hatte man noch mit begrenzten Speichergrössen und langen Verarbeitungszeiten zu kämpfen. Die Folge waren Programmierkunststücke auf der untersten Datenebene, den sogenannten Bitstrings.

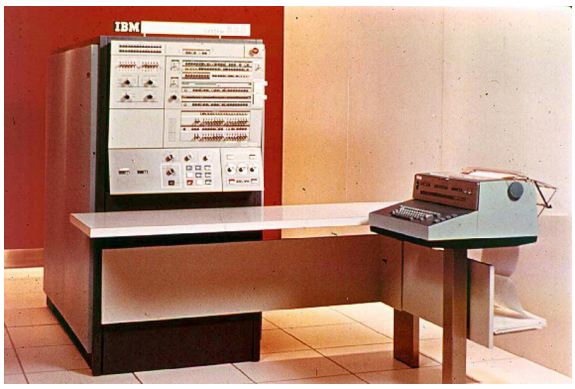
Über eine Anweisungstabelle wurde dem Computerprogramm ADSS vorgegeben, wie und welche Daten miteinander in Beziehung zu setzen waren. Dadurch wurde die Grundlage für spätere Auswertungen geschaffen.

Teilweise erforderte die Situation der Datenquellen, dass diese zunächst in ein technisch verarbeitbares Format überführt werden mussten. Dabei war viel Handarbeit gefragt. Trotzdem gelang es mir, ADSS betriebsbereit zu programmieren und mit immer grösseren Datenmengen nächtliche Tests durchzuführen.

Die Verarbeitung sämtlicher „Livedaten“ dauerte auf dem Grossrechnersystem jeweils mehrere Stunden. Fast unvermeidlich versteckte sich unter den Hunderten von PL/I-Anweisungen leider auch ein Programmierfehler. Dieser hatte zur Folge, dass das Grossrechnersystem während der nächtlichen Verarbeitung jeweils abstürzte und dadurch auch andere Anwendungen nicht mehr ausgeführt werden konnten. Da die Verarbeitungen nachts unbeaufsichtigt abliefen, waren am Morgen sämtliche Arbeiten unerledigt. Nach mehreren nächtlichen Abstürzen, ohne dass der Verursacher bekannt war, wurde mir vorsorglich ein Testverbot auf dem Grossrechnersystem auferlegt.

Spätere Untersuchungen ergaben, dass ADSS zusammen mit einem Fehler im Betriebssystem zu diesen „Katastrophen“ geführt hatte. Eine kurze Erklärung für Informatiker sei mir an dieser Stelle erlaubt: Eine falsche Adresszuweisung innerhalb von ADSS veränderte Daten im S/360-Betriebssystem – etwas, das eigentlich gar nicht hätte möglich sein dürfen. Die Folge war ein Verbot weiterer Tests und damit letztlich das Ende von ADSS.

Übrig geblieben von diesem Projekt sind die nachfolgenden technischen Erklärungen und Datendefinitionen (ohne Gewähr auf Vollständigkeit).



Quelle: static.righto.com/images: Photo source unknown

* * * * *

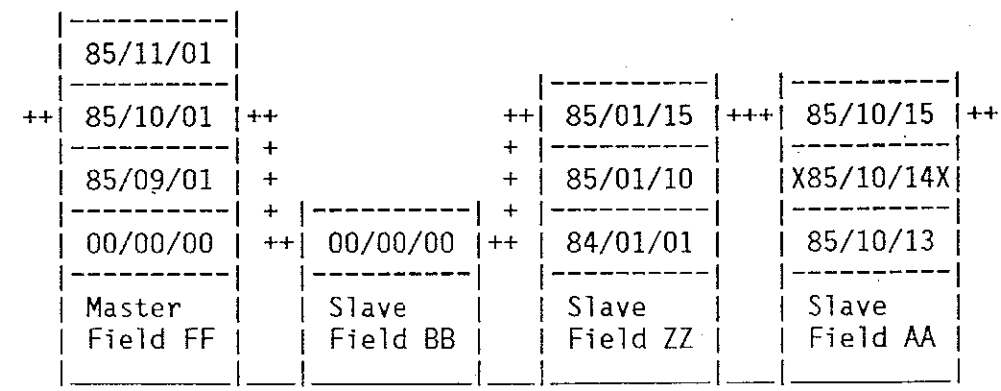
All Direction Search System (ADSS)
****

24.03.87/STAL
DRAFT/S501020

ADSS ist ein erweitertes Data-Dictionary-System.
Die Data Dictionary System Working Party der British Computer Society bezeichnet einen Data Dictionary als Werkzeug zur Verwaltung von Strukturen und Daten die Daten beschreiben.

ADSS ist faehig die unterschiedlichsten Daten zu speichern, Veraenderungen chronologisch festzuhalten und Beziehungen zwischen den Datenfelder zum Zwecke von Untersuchungen automatisch zu bilden und durch Verwendung einfacher Commands zusammenhaenge aufzulisten.

$DATE = LAST (or $DATE=)



++ Selected item(s)

All Direction Search System (ADSS) 24.03.87/STAL
********** DRAFT/S501020

ADSS ist ein erweitertes Data-Dictionary-System.
Die Data Dictionary System Working Party der British Computer Society bezeichnet einen Data Dictionary als Werkzeug zur Verwaltung von Strukturen und Daten die Daten beschreiben.

ADSS ist faehig die unterschiedlichsten Daten zu speichern, Veraenderungen chronologisch festzuhalten und Beziehungen zwischen den Datenfelder zum Zwecke von Untersuchungen automatisch zu bilden und durch Verwendung einfacher Commands Zusammenhaenge aufzulisten.

ADSS kann sich gegenueber anderen Data Dictionary Systemen in folgenden Punkten unterscheiden:

Inquiry

- Element- und Datenbezognes inquiry unterstuetzt.
- Online Inquiry aller gespeicherten Daten und ihrer Zusammenhaenge.
- Veraenderungen koennen zeitlich zurueckverfolgt werden.
- Persoenliche VIEWS durch dynamisches unterdruecken von Lade-Gruppen (RACF-GROUPS).
- Batch-Inquiry unterstuetzt.
- DB-Load-Summary fuer LOADED-GROUPS jederzeit ersichtlich (selective gemaess RACF-LU)

Allgemeines

- Fuer grosse Datenmengen konzipiert (> 100 MB).
- Datenverbindungen (implosion, explosion) werden aufgrund gleicher Datenwerte automatisch hergestellt.
- Keine vorgegebene Datenstruktur.
- Automatische Zwei-stufen-hirarchie (Parent/Children).
- Keine Elementbezogenen Daten-Banken oder sonstige vorgegebenen Definitionen.
- ADSS kann aus mehreren DD-System bestehen (Test, Produktion, etc., separate Link-Records, aber gemeinsame Daten-Wert-DB).
- Keine redundanten Basisdaten innerhalb saemtlicher angegliederten Applikationen und DD-Systemen.

./.

- Data links (key links) koennen beim laden der Daten jederzeit erweitert werden.

- Additional Text-Daten-Speicherung moeglich.

- 255 Lade-Groups (resp. Applikationen) unterstuetzt.

- 110 Feldnamen pro Group moeglich.

- 16 Feldnamen pro Application-Load-Job moeglich.

- 1024 Feldnamen, tabellen gesteuert, synonym und alias unterstuetzt.

- Standart Daten-Lade-Program (Input-Conversion-utility).

- Access-Statistics verfuegbar.

Security

- Erhoehte Sicherheit bezueglich Fremdzugriffe durch unzusammenhaengende Datenspeicherung.

- Die externe und interne Security ist mit RACF realisiert.

- Die interne Security erlaubt selektive VIEWS (RACF internal LU).

- Zugriff auf die Link-DB nur durch authorisiertes PGM (RACF).

- Interne RACF-GROUPS koennen beim Laden definiert werden (unabhaegnig von bestehenden RACF-GROUPS).

- AUDIT-special-access (all fields), wenn der Benuetzer das Racf-AUDIT attribut hat.

Ungeignete Daten sind:

- Resultate von Berechnungen (koennen aber als Text in der ADSS-Text-DB gespeichert werden).

- Bereits konsolidierte Daten (nur Source-Daten verwenden).

Vereinfachter technischer Ueberblick

Die Datenspeicherung erfolgt getrennt nach Feldnamen, in einer VSAM KSDS-Datenbank.

Der DB-Key besteht aus den Werten der einzelnen Felder (einer Ladestruktur). Jeder Wert kann nur einmal vorkommen (UNIQUE-KEY).

Der DB-Datenteil besteht aus einer fortlaufenden Nummer welche wiederum Key einer Datenbank, mit den Feldverbindungen, ist.

Die DB mit den Feldverbindungen beinhaltet alle Verbindungen (links) zu den zugehoerigen Feldern des gleichen Laderecords.

Feldnamen werden wie Daten behandelt. Es kann daher mit einem Daten-Wert, sowie einem Daten-Feldnamen auf die DB zugegriffen werden.

Alle Records mit Feldern gleichen Namens und Inhaltes bilden automatisch eine Verbindung (gemaess Input-Lade-Definitionen koennen weitere Verbindungen hergestellt resp. unterdrueckt werden).

Aufgrund von speziellen Vorkehrungen koennen die geladenen Records wieder zu ihrer Gesamtheit rekonstruiert werden.

Inhalt

AA. Technische Voraussetzungen

TSO / ISPF
VSAM
RACF 1.7

BB. Uebersicht

01. Ablaufuebersicht

02. Datenbanken / Files

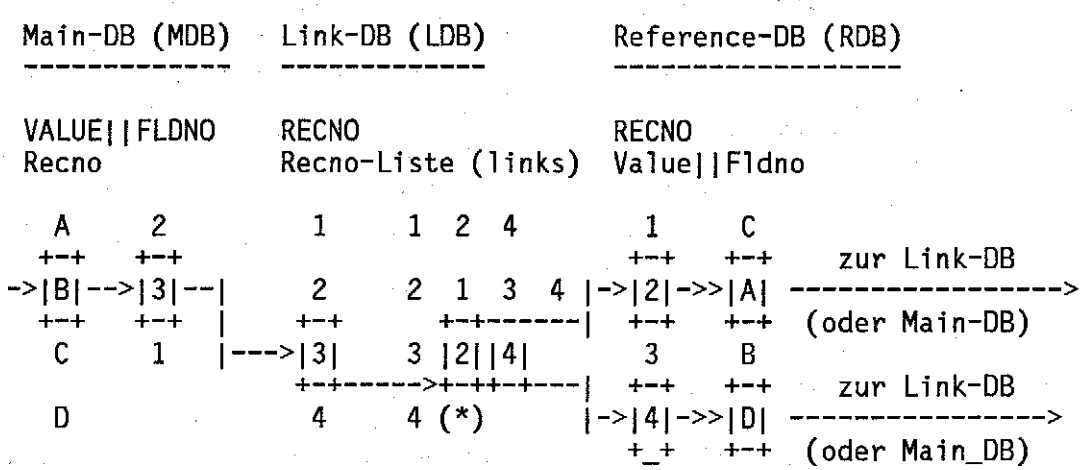
01. Ablaufuebersicht

Der Einstieg erfolgt durch die Angabe eines Wertes und ev. eines Feld-Namens oder eines Feld-Namens und ev. eines Wertes.

Access-logic by value.

Die Link-Nummer auf der Main-DB zeigt auf den Record in der Link-DB, wo saemtlichen Datenverbindungen dieses Wertes festgehalten ~~ist.~~ sind.

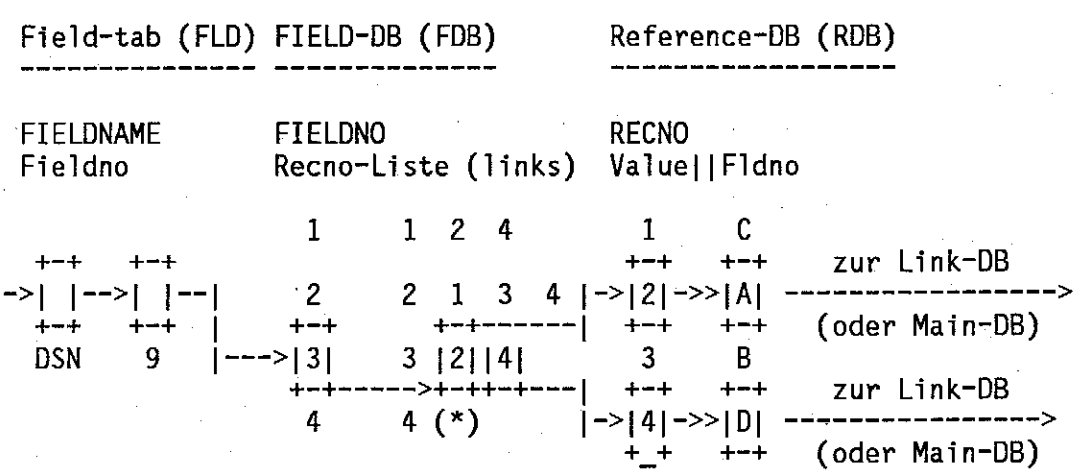
Die Werte aller dieser Verbindungen sind in der Reference-DB enthalten. Diese Werte koennten wiederum fuer einen Zugriff auf die Main-DB verwendet werden (als Einstieg fuer sequentiellen Zugriff).



Das erste Element der Link-DB zeigt immer auf sich selbst (Feld-Beschreibung: Datum/Feldno/Linkno).

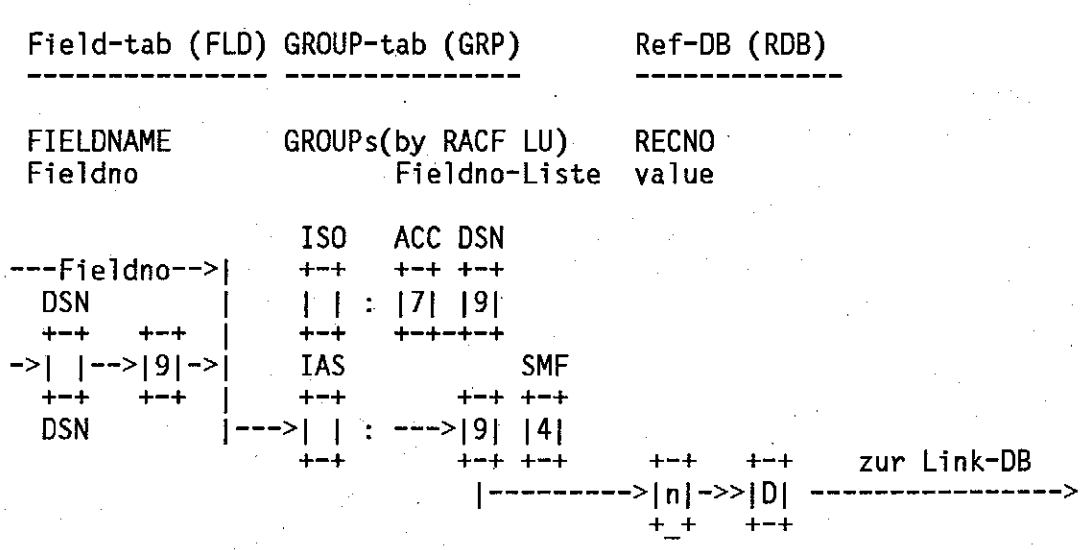
(*) Constant-Data - keine 'Backward-Reference' verfuegbar, ausser die Daten wurden beim laden mit der Option "Constant-Backward-Link" markiert.

Access-logic by field-name.



|| ≙ Concatination of value

View-selection logic (security) by RACF-GROUPS



02. Datenbanken / Files (DD-names)

Bezeichnung	Organisation	Inhalt (Uppercase=KEY).
Main-DB (ADSSMDB)	VSAM/KSDS	Recno, FIELD-VALUE || FIELDNO
Reference-DB (ADSSRDB)	VSAM/KSDS	RECNO, Field-Value || Fieldno
Link-DB (ADSSLDB)	VSAM/KSDS	RECNO, |: Load-Date, Status, Groupno, Fieldno, Recno :|
Field-Search-Table (ADSSFDB)	VSAM/KSDS	FIELDNO, Fieldcnt, | :Recno :|
Text-DB (ADSSTDB)	VSAM/KSDS	FIELDNO || RECNO || DSTAMP || LCOUNT, Text
Fieldref-File (ADSSFLD)	PS	FIELDNO, Fieldname or Alias, Truename Shortname, Fieldno, Description
Group/Field-Link (ADSSGRP)	PS	GROUPNO, Group, |: Fieldno :|
Control-file (ADSSCTL)	PS	Applic, Error, Testno, Startno, Stopno, Started, Ended, Hrecno, Freecnt, Freeno(32767)
Log-File (ADSSLOG)	PS	APPLIC, TESTNO, STARTED, Ended, Runtime, Recordnumbers and add. infos
Procedure-File (ADSSPRO)	PO	ADSS Command-Procedures
Error-MSG-File (ADSSMSG)	PS	Inquiry Error-Messagefile
Report-File (ADSSREP)	PS	Inquiry Outputfile
Summary-File (ADSSSUM)	PS	Inquiry Trace-Outputfile (Summary)

Record-Struktur

Bezeichnung	Key (length in byte)	Data (length in byte)	Bemerkungen
ADSSIN			ADSS-Input
	Tstamp (10)		YYMMDDnnnn
			(0=Constant)
	Groupname (8)		Group(source)
		Mfield (8+2) or	Masterfield
		Sfield (8+2)	Slavefield
		Opcode (3)	Oper-Code
		Retain (3)	Retainevalue
		Value (45)	Field-Value
ADSSMDB			Main-DB
		Recno (4)/bin fix 31	LDB-Linkno
	Value (45)		Field-Value
	Fieldno (2)/bin fix		Fieldnumber
ADSSLDB			Cross-Link-DB
(spanned)	Recno (4)/bin fix 31		MDB-Linkno
	Testind(2)/ b. fix15		Test/Prod ind
	Reccnt (2)/ b. fix15		Cont-count
		Groupno (4)/pic'zzzz'	rec. owner
		Limit (4)/b. fix 31	Type (len 8)
		Linkcnt (4)/b. fix 31	Total linkcnt
		Lnkalen (4)/b. fix 31	argum-length
		Linflen (4)/b. fix 31	funct-length
		Linktab(3x32752,VAR)	Value-array
		(3272 x 80 Bit x 10)	32720 Elem.
		Dstamp 32 Bit(bin)	date yymmdd
			-1000000 rec,
			sonst extern.
		Group 08 Bit	Group(max255)
		Fieldno 16 Bit	Fieldnumber
		Status 1 Bit	0=ACT / 1=DEL
		Textdb 1 Bit	Access TDB
		Direct 1 Bit	No DB-Access
		Numeric 1 Bit	Direct+numer.
		Reserve 4 Bit	
		Recno 24 Bit(bin)	RDB-Linkno or
			CHAR(3) or
			Numind 1 Bit
			Mask 3 Bit,
			Numer. 20 Bit
			(RECNO x -1
			is text)
		>sequential, rolling<	

ADSSRDB	Recno (4)/bin fix 31	Value (47) var (Data-value (45) Fieldno(2)/bin fix 15)	Reference-DB MDB-Linkno Value-field MDB-key-value Fieldnumber
ADSSFLD		Fieldno (04,num pic) Fieldnam (08) Truename (08) Shortnam (08) Descr (30)	Field-descr Fieldnumber Fieldname or ALIAS Eff. name for ALIAS Short-Name Description
ADSSFDB	Fieldno(2)/b. fix 15 Testind(2)/b. fix 15 Reccnt (2)/b. fix 15	Fieldcnt(4) b. fix 31 rectab(10916). Recno (3)/ bit 24 (bin fix 31)	Field-search Field-number Test/Prod ind Cont-count Field-count RDB-Linkno
ADSSGRP	Groupno (4,num pic)	Group (8) Udtedate (8) Fieldtab(255), Fieldno (4, num pic)	Field-Table GROUP-number Record-No Last-loaddte Array Field-number
ADSSTDB	Fieldno (2)/ b. f.15 Recno (4)/ b. f.31 Dstamp (4)/ b. f.31 Testind (2)/ b. f.15 Lcount (2)/ b. f.15	Text (150) var	Text-DB Field-no LDB-recno Time-stamp Test-ind Line-count Text

ADSSCTL			ADSS-CONTROL
		Applic (20)	Appl/Jobnr
		Error (1)	Errorind(Y/N)
		Testno (4) dec f. 7	Test-number
		Startno(4) dec f. 7	Inp-start-no
		Stopno (4) dec f. 7	Inp-stop-no
		Started(15)	Date/Time
		Ended (15)	Date/Time
		Hrecno (4, dec fix 7)	Highest RECN
		Freecnt(4, dec fix 7)	No free elem.
		Freeno(32767) (4, dec fix 7)	Free RECNO
ADSSLOG (WRITE/ REWRITE)	Applic (20)		ADSS-CONTROL
	Testno (2) pic		Appl/Jobnr
	Started(15)		Test-number
			Date/Time
		Ended (15)	Date/Time
		Startno(7) pic	Inp-start-no
		Stopno (7) pic	Inp-stop-no
		Runtime(7) pic	Runtime

CC. Input-description

Record-Type

Col. 19:

```
M     = Master-record (Unique-key)
S     = Slave-record's (Unique-keys)
```

Master-Record (* col. 19)

ADSSIN.Opcode:

Operation-descriptions col. 30-32.

Record-description

```
 ______ Operation-code (date related)
|  ____ Value related: Blank - Nocheck
| |                    N=New - value of field(s) must not yet exist
| |                    O=Old - value of field(s) must exist
| |  __ C = add slave only if value of equal field(s) has CHANGED
| | |  A = Add slave according to the slave-opration-code.
| | |
| | |
A . . = Add master (leave last-update-date unchanged)
R . . = Replace master (add if not found / update last-update-date)
D(O). = Delete - flags field as deleted, history available.
E(O). = Erase - no history available.
```

Col. 34 - Retain

```
nnn   = Number of kept links
blank = rolling (default 4031 links)
```

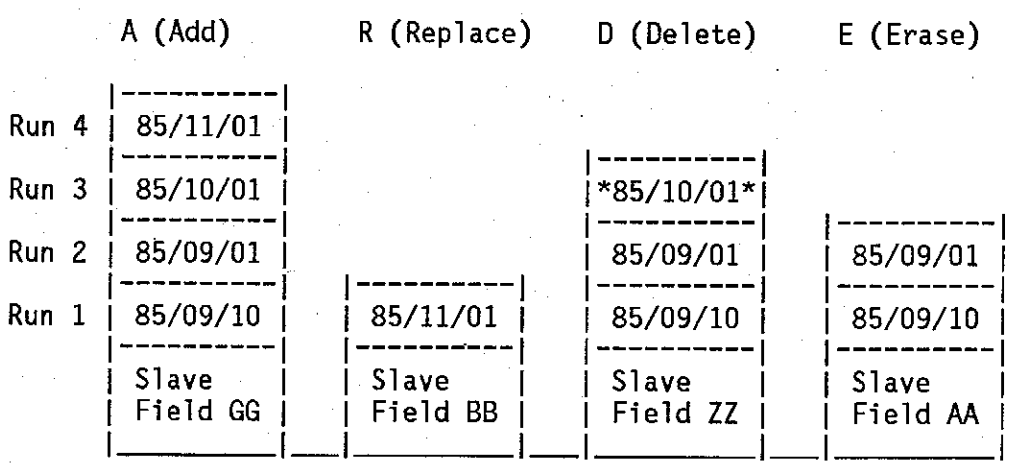
Field-description

Slave-Record (Cross-link-definition,
according to master-opcode-definition (pos. 3 A or C)).

Col. 30:

A	= Add pointer-segment
R	= Replace pointer-segment
D	= Delete pointer-segment (set delete flag)
E	= Erase pointer-segment (remove pointer-segment)

Example user view (col. 30):



Col. 31:

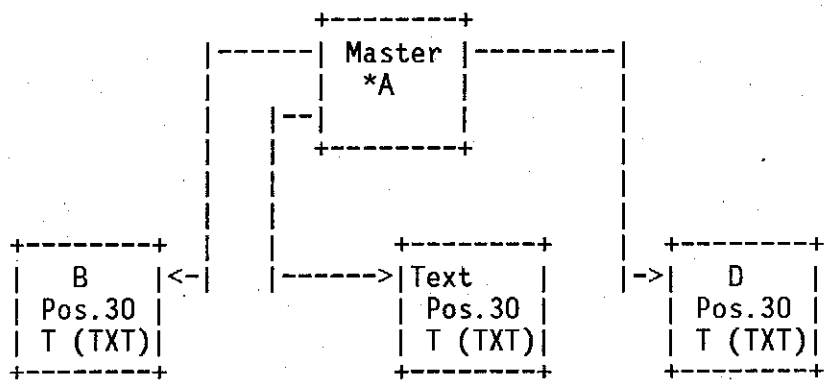
T	= Text (no backward references available).
N	= No backward references to establish.
S	= Short backward-link (to master-field only).
F	= Full backward link, including sub-field-cross-references.

T- and N-related-values may be stored as direct value, if
the value does not exceed Char 3 Byte or 8388608 numerically
(+ or -).

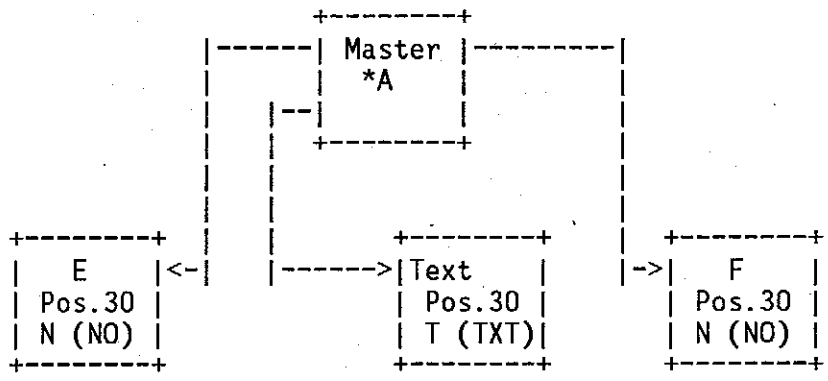
DD. Field-Record Data-Cross-Link-Definition-Description.

Input-pos. 30:

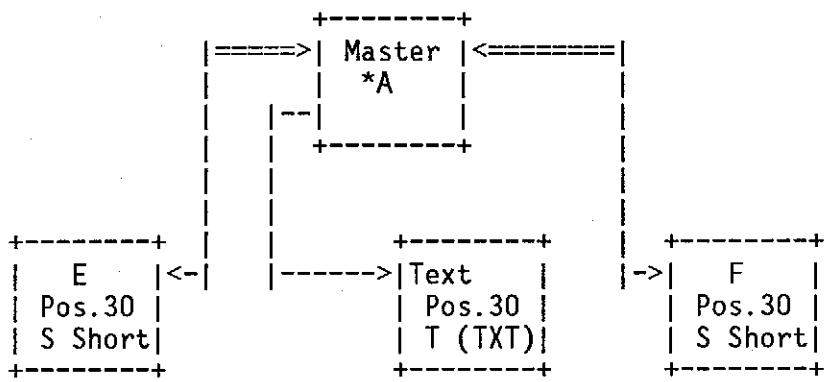
T = Text (no backward references available).



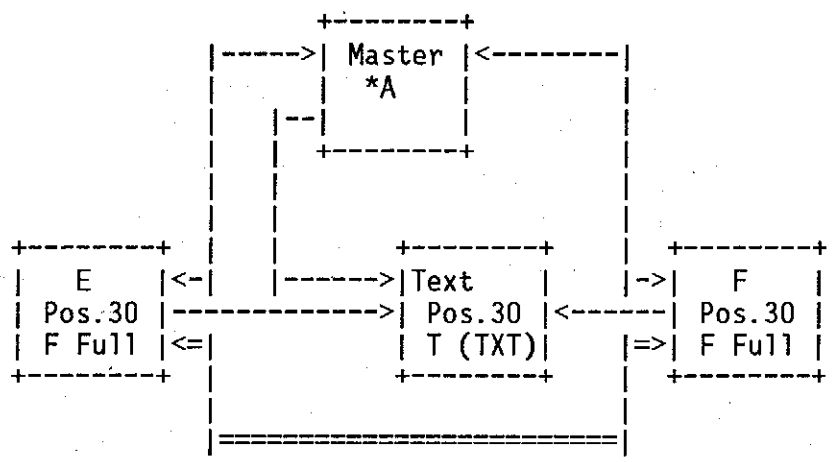
N = No backward references to establish.



S = Short backward-link (to master-field only).



F = Full backward link, including sub-field-cross-references.



EE. Batch reformat-program (adsspre)

SYSIN variables:

```
1 INPDEF,
  2 GROUP   CHAR(8),
  2 START DEC FIXED (7),
  2 STOP  DEC FIXED (7),
  2 SELECT CHAR(45),
  2 POS    DEC FIXED(3),
  2 LENGTH DEC FIXED(3),
  2 MASTER,
    3 FLDNAME CHAR(8),
    3 OPCODE CHAR(3),
    3 DATA    CHAR(45),
    3 RETAIN PIC'ZZZ',
    3 POS    DEC FIXED(3),
    3 LENGTH DEC FIXED(3),
  2 SLAVE(15),
    3 FLDNAME CHAR(8),
    3 ZEROBLK CHAR(1),
    3 OPCODE  CHAR(2),
    3 DATA    CHAR(45),
    3 POS     DEC FIXED(3),
    3 LENGTH  DEC FIXED(3);
```

SYSIN variable-example:

```
//SYSIN  DD *

 INPDEF.GROUP           = 'IAS     '
 INPDEF.SELECT          = '  DASD'
 INPDEF.STOP            = 999999
 INPDEF.POS             = 150
 INPDEF.LENGTH          = 6


 INPDEF.MASTER.FLDNAME = 'DSN'
 INPDEF.MASTER.OPCODE  = 'A C'
 INPDEF.MASTER.RETAIN  = 365
 INPDEF.MASTER.POS     = 2
 INPDEF.MASTER.LENGTH  = 44


 INPDEF.SLAVE.FLDNAME(01) = 'RESOURCE'
 INPDEF.SLAVE.OPCODE (01) = 'AS'
 INPDEF.SLAVE.DATA   (01) = 'DASD'
 INPDEF.SLAVE.POS    (01) = 0
 INPDEF.SLAVE.LENGTH (01) = 0

 INPDEF.SLAVE.FLDNAME(02) = 'SMFTYPE '
 INPDEF.SLAVE.OPCODE (02) = 'RT'
 INPDEF.SLAVE.POS    (02) = 75,
 INPDEF.SLAVE.LENGTH (02) = 2,

 INPDEF.SLAVE.FLDNAME(03) = 'VOLUME  '
 INPDEF.SLAVE.OPCODE (03) = 'AS'
 INPDEF.SLAVE.POS    (03) = 51
 INPDEF.SLAVE.LENGTH (03) = 6

 INPDEF.SLAVE.FLDNAME(04) = 'LABEL   '
 INPDEF.SLAVE.OPCODE (04) = 'AT'
 INPDEF.SLAVE.POS    (04) = 57
 INPDEF.SLAVE.LENGTH (04) = 3

 INPDEF.SLAVE.FLDNAME(05) = 'VOLCNT  '
 INPDEF.SLAVE.ZEROBLK(05) = 'Y'
 INPDEF.SLAVE.OPCODE (05) = 'RN'
 INPDEF.SLAVE.POS    (05) = 49
 INPDEF.SLAVE.LENGTH (05) = 1
```

Batch-Input-Example

```
....+....1....+....2....+....3....+....4....+....5.../ /...8

|-batch-load-key-|

8510150910RACF    MDSN        A C 62ISO.STAL.* (G)
8510150910RACF    SUACC       AS    NONE
8510150910RACF    SOWNER      RS    S082163
8510150910RACF    SAUDIT      AS    FAILURES(READ)

8510151622SMF     MVOLUME     ANA   DS0637
8510151622SMF     SVOLSEQU    RT    1
8510151622SMF     SLABEL      RT    5
8510151622SMF     SRESOURCE   RS    TAPE
8510151622SMF     SDSN        RT    WTAAS.DB.SAVE
8510151622SMF     SSMFTYPE    RN    14
8510151622SMF     SSMFDATE    RN    851014
8510151622SMF     STIME       RN    230817

8510151623SMF     MVOLUME     A A 10DS0637
8510151623SMF     SVOLSEQU    RN    1
8510151623SMF     SLABEL      RN    1
8510151623SMF     SRESOURCE   RS    TAPE
8510151623SMF     SDSN        RS    MA.TEMP
8510151623SMF     SSMFTYPE    RT    15
8510151623SMF     SSMFDATE    RS    851014
8510151623SMF     STIME       RT    230817

8510150001PDS     MPERSNR     ACA   0076823
8510150001PDS     SNAME       RS    HUGELSHOFER
8510150001PDS     SFNAME      RS    HANS
8510150001PDS     SLOCATION   RS    HOPB
8510150001PDS     SSIGN       RS    HUGH
8510150001PDS     SMANAGER    RS    0041312
```

Preceeding blanks on the accuracy-field will be removed.

Att: The header-field-name must not be repeated within the depended records (field-accurancy has to be build).

For non-constant-values following rules appears:

Value will be truncated at first blank-sign.

Following special-signs will be translated to ".", as:

/ - + : ; ? ! ¬ = < > &

Following special-signs will be removed from input:

" ' ( ) % *

FF. SPF-Screens

ADSS inquiry is a command driven application.

All output is displayd with SPF BROWSE / EDIT screens, including a parm-display-line, showing the current value of several parameters, as PAGE 66, TABS ON, etc.

Additional screens are:

-Start screen (with basic ADSS-instructions)

-Help screens

-ADSS-Command-generater-screen.

-ADSS-Command-Procedure-generater-screen.

GG. Inquiry notation and examples (Direct- und Proc-Language).

On command-line (CMD===>) enter:

//.... ADSS commands

/..... direct search commands

//.... (ADSS commands):

Any commands can be entered without ( ) - defaults will be showed.

```
//HELP
//REPORT                                &sysuid.&syst.ADSSREP
//SUMMARY                               &sysuid.&syst.ADSSPRT
//TABULATOR (ON | OFF | MIX)            OFF
//TABSET    (membername)                &sysuid.&syst.ADSSCTL($TABSET)
//TRACE     (ON | OFF)                  OFF
//PAGE      (nn)                        66
//ACCESS    (n)                         1 (min.)
//LOCATE    (value)                     blank
//EDIT                                  dsn last showed
//BROWSE                                dsn last showed
//RETRIEVE  (n | 1 to 9)                1
//TITEL     (membername)                &sysuid.&syst.ADSSCTL($TITEL)
//CLASS     (IUO | IC | ICR | RIC)      IC
//PROCEDURE (membername)                &sysuid.&syst.ADSSPRO(mbrname)
//STOREFILE (dataset.name)              &sysuid.&syst.ADSSREP
//SAVEREP   (dataset.name)              any valid dsn
//LOGNAME   (dataset.name)              &sysuid.&syst.ADSSLOG
//LOGLIST                               logname.dataset.name
//SUB       (membername)                &sysuid.&syst.ADDSCTL($SUB)
//DEFAULTS                              &sysuid.&syst.ADSSCTL($DEFAULT)
//GROUPS                                &sysuid.&syst.ADSSGRP
//FIELDS                                &sysuid.&syst.ADSSFLD
//GRPEXCL (group_x, .... | ANYGRP*)     () = all GROUPS inactive
//GRPINCL (group_y, .... | ANYGRP*)     () = all GROUPS active
```

/.... (Direct search commands):

```
/fieldname <opcode> <value> <,>,<fieldname <opcode> <value>,<,><...>>
           =                          ,<fieldname (fieldname <,> fieldname)>
           =<
           >=
           <>
```

Fieldnames must be separated by commas, redundant commas will be embeded in result-record.

```
/$..... <<opcode> <value> ...>


$DATE <opcode> <LAST | ALL | FIRST | yymmdd | TODAY | PREV | FUTURE |
              OLDEST>
$PAGE                                  * new page before printing
$FIELDS  (RECORD | EXTERNAL | ALL)     * field-link-type selektion
$VALUE   <(opcode generic)>            * fieldvalue selection
$NUMBER  <(opcode nnnn)>               * record-number range
$INCLCMD (membername)                  * include a adss cmd-procedure
$INCLTXT (membername)                  * include any text
$SKIP    <(n)>                         * skip before printing
$ILVL                                  * current instructionlevel (nn)
$COMPARE <(DIFF | EQUAL)>              * list changed, nochanged items
fieldname ($DELETED <opcode $DATE>)    * list items with delete-flag
```

Examples:

```
CMD===> /CLASSIF = ICR,, DSN,, OWNER (FNAME, TELNO,, MANAGER, (TELNO)
```

Output (TABS OFF):

```
        ICR, IBM.PAYROLL, MEIER HANS 3451, HUBER, 6712
        ICR, IBM.MARIS, FISCHER FRITZ 2265, MUELLER, 5729
```

Current TAB-member:

```
      _*__._*__1____.____2___*.____3____*____4_*__.__*_5____.___*6_
```

Output (TABS ON):

```
      -CLAS DSN                OWNER      FNAME   TELNO MANAGER     TELN

       ICR, IBM.PAYROLL,       MEIER      HANS    3451, HUBER,      6712
       ICR, IBM.MARIS,         FISCHER    FRITZ   2265, MUELLER,    5729
```

Direct-procedure:

```
   /$PAGE -
    MANAGER =< MUELLER (FNAME, -
    $SKIP(2),, -
    $DATE = TODAYS, -
    EMPLOYEE (FNAME))
```

Other examples:

```
/TAPE                          - displays all tape-volumes.

/TAPE(LABEL)                   - displays all tape-volumes and labels.

/TAPE($VALUE,$FIELDS,$DATE)    - displays all available informations.

/TAPE($)                       - displays all available informations.

/TAPE($,$D=)                   - displays all available informations
                                 about $DATE=LAST.
```

Access Statistics (summary)

```
DATE: 28.02.1987
RECORDS TO PROCESS LIMITED TO:   100
INSTR-IDENT: PERSNO1  STARTED: 08.09  ENDED: 08.09 RUNTIME: 0 FULL-MIN
 ILVL    NT    NO    CNT   COMMAND-INSTRUCTION

   1     1     1     14   #RUNCTRL  ADSSFDB,PERSNO  * PERSONAL-NUMBER
   1     1     2     14    DO
   1     2     3     14     WRITE     002,N
   1     2     4     28     READ      ADSSLDB,NAME
   1     2     5     14     COMPARE   $NUMBER,EQ,1
   1     3     6     14      WRITE     022,L
   1     2     7     28     READ      ADSSLDB,FNAME
   1     2     8      0     COMPARE   $NUMBER,EQ,1
   1     3     9      0      WRITE     044,L
   1     1    10     14    END
   1     1    11     14   #RUNCTRL  END
```

Procedure-Language.

The lists of extensions are not complete, see direct commands.
The search-mechanism depends on the load technic used for input
(relational, hirachical, redundant or any combinations of those).

Commands/Description

```
*         Comment-line

any_cmd   Extension    * any comment can be added after asterisk

#RUNCTRL  Start/end procedur
READ      Read DB or any other ADSS-procedure
COMPARE   Compare value
DO        Begin DO-group
STORE     Store value
END       End of DO-group
ELSE      Reverse true-condition
SORT      Sort selected records
WRITE     Write out value
```

Commands/Notation

```
*         any comment can be added after asterisk

#RUNCTRL  ADSSFDB,fieldname ',SCAN'
          ADSSMDB,value* ',SCAN'
          END

READ      ADSSLDB,fieldname '
          ADSSMDB,fieldname '
          ADSSLDB,$DATE '
          ADSSLDB,$FIELD '
          ADSSLDB,$PROC(member_name)
          ADSSxDB,.........,op,.......... * see COMPARE

COMPARE   $NUMBER,op,number
          $FIELD,op,posnr,length,value
          $VALUE,op,posnr,length,value
          $VALUE,op,posnr,length,$BLANK
          $DATE,op,posnr,length,yymmdd
          $DATE,op,,,$LAST
          $DATE,op,,,$FIRST
          $NUMBER,op,number
          $STORE(dim_x,dim_y),op,posnr,length,......

DO

STORE     dim_x,dim_y,$VALUE
          dim_x,dim_y,value
          dim_x,dim_y,$DATE
          $ILVL,dim_y,$VALUE

END
```

Example:

List all userid's of Meier Hans, HO*

a) The input-laod was relational without any redundance key-links).

```
STORE *,*,$BLANK
#RUNCTRL ADSSFDB,NAME = MEIER
 DO
  STORE $ILVL,1,$VALUE
  READ ADSSLDB,PERSNO * relational link with all pers-values
   DO
    STORE $ILVL,2,$BLANK
    STORE $ILVL,3,$BLANK
    READ ADSSLDB,FNAME = HANS
     STORE $ILVL,2,$VALUE
    READ ADSSLDB,LOCATION = HO*
     STORE $ILVL,3,$VALUE
   END
   COMPARE $STORE($ILVL,1&2&3),NE,,,$BLANK
    DO
     READ ADSSLDB,USERID
      DO
       WRITE $STORE($ILVL,1),4
       WRITE $STORE($ILVL,2),16,L
       WRITE $STORE($ILVL,3),22,L
       WRITE $VALUE,4,N
      END
    END
 END
#RUNCTRL END
```

```
ELSE

SORT     Sort definitions (according to sort-standards)

WRITE    position     * Previous value is repeated on each line
         position,L   * Previous value is not repeated
         position,N   * Skip to new line before writing (not repeated)
         $STORE(dim_x,dim_y),.....
         $STORE($ILVL,dim_y),.....
```

GG. Browse/Save output-selection example

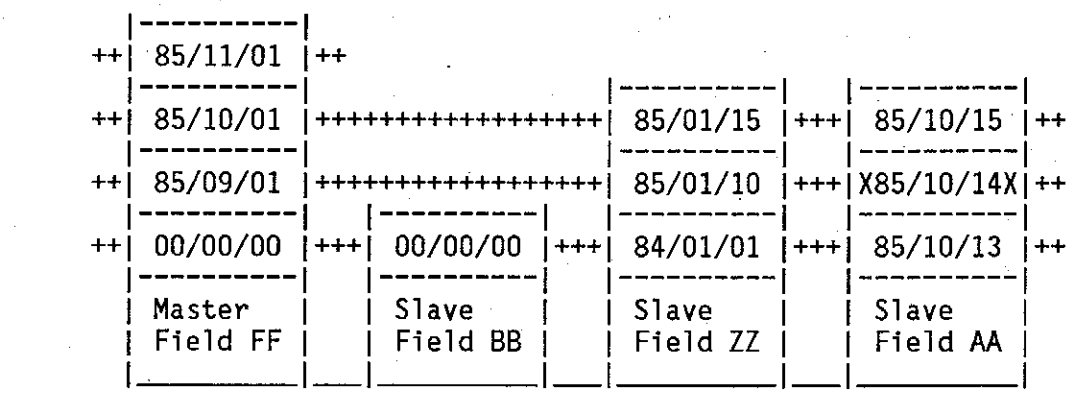
Master selection example with use of
2nd byte (assume current date is 85/10/15)

Date 00/00/00-fields are constant-data-elements.

Common-values consists of:

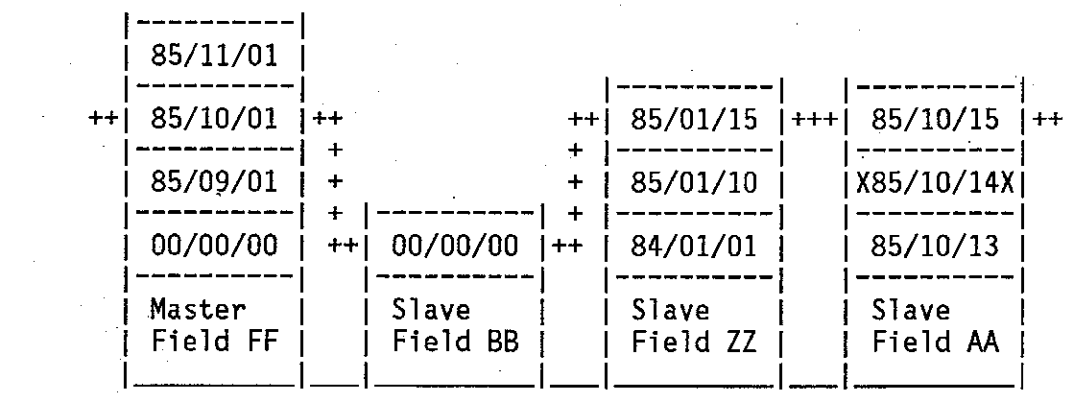
Fieldname / Date-stamp and only one value/date possible.

$DATE = ALL (or $DATE)



++ Selected item(s)

$DATE = LAST (or $DATE=)

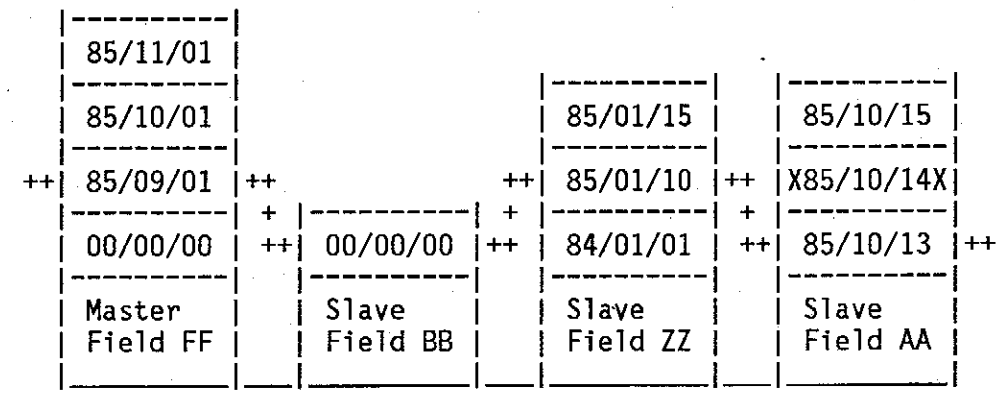


++ Selected item(s)

b) The input-load was search-access-related.

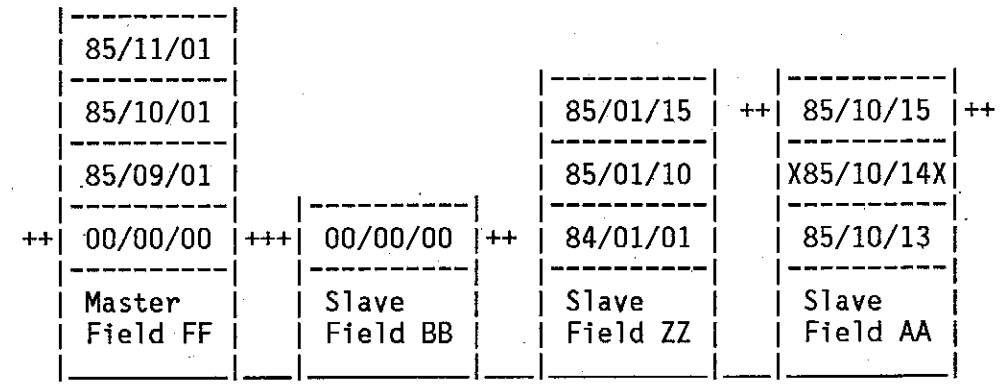
```
#RUNCTRL ADSSFDB,NAME = MEIER
 DO
  STORE *,*,$BLANK
  STORE $ILVL,1,$VALUE
  READ ADSSLDB,FNAME = HANS
    STORE $ILVL,2,$VALUE
  READ ADSSLDB,LOCATION = HO*
    STORE $ILVL,3,$VALUE
  COMPARE $STORE($ILVL,1&2&3),NE,,,$BLANK
   DO
    READ ADSSLDB,USERID
     DO
      WRITE $STORE($ILVL,1),4
      WRITE $STORE($ILVL,2),16,L
      WRITE $STORE($ILVL,3),22,L
      WRITE $VALUE,4,N
     END
   END
 END
#RUNCTRL END
```

$DATE = PREVIOUS



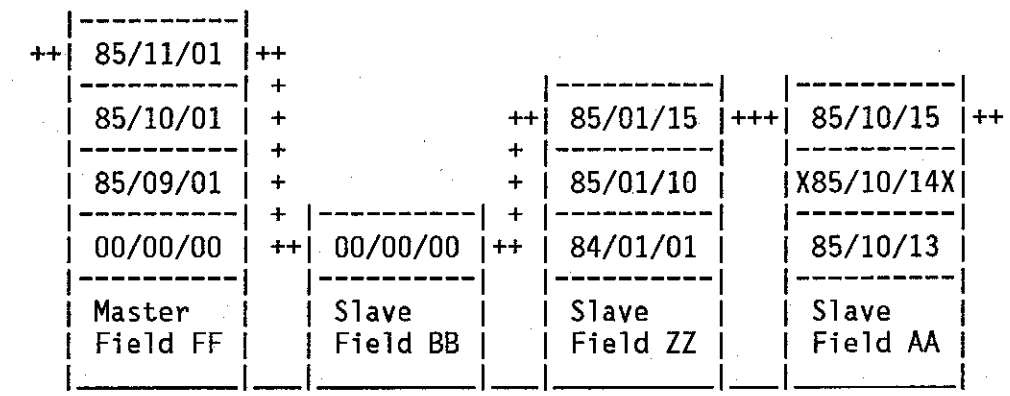
++ Selected item(s)

$DATE = TODAYS (incl. constant-values, date 00/00/00)



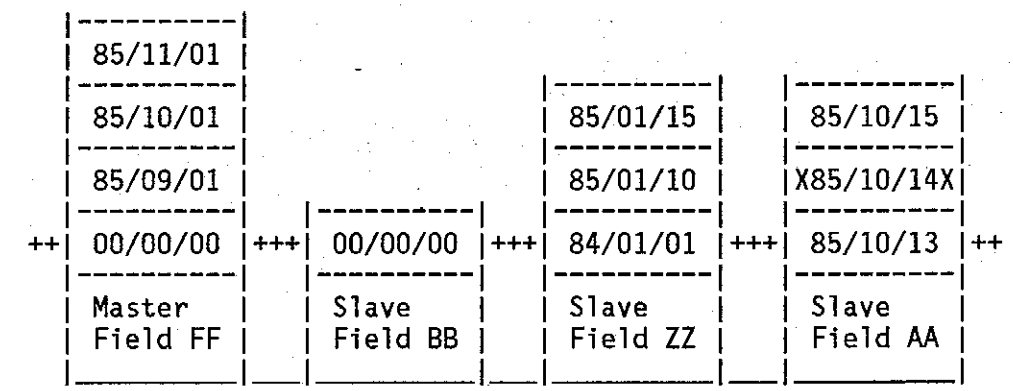
++ Selected item(s)

$DATE = FUTURE



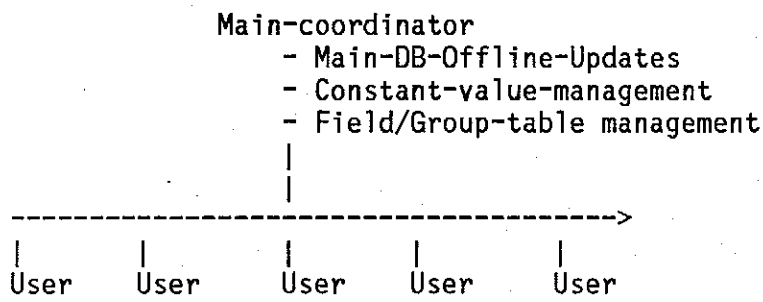
++ Selected item(s)

$DATE = OLDEST



++ Selected item(s)

HH. Admin organisation



JJ. How to develop an ADSS-application

1. Get a list with all constant-values from the main-coordinator.

2. Design the final list you want (fields/subfields).

3. How do you access? By fieldnames or vaues?

4. Get clear what you need: Fields/Subfields. Where do they come from (which application (the group it belongs to)).

5. Make a chart with all field-relations you need (you must now know your key fields). See DD.

6. Try to write an input example considering all constant-values and cross-link-rules. See EE.

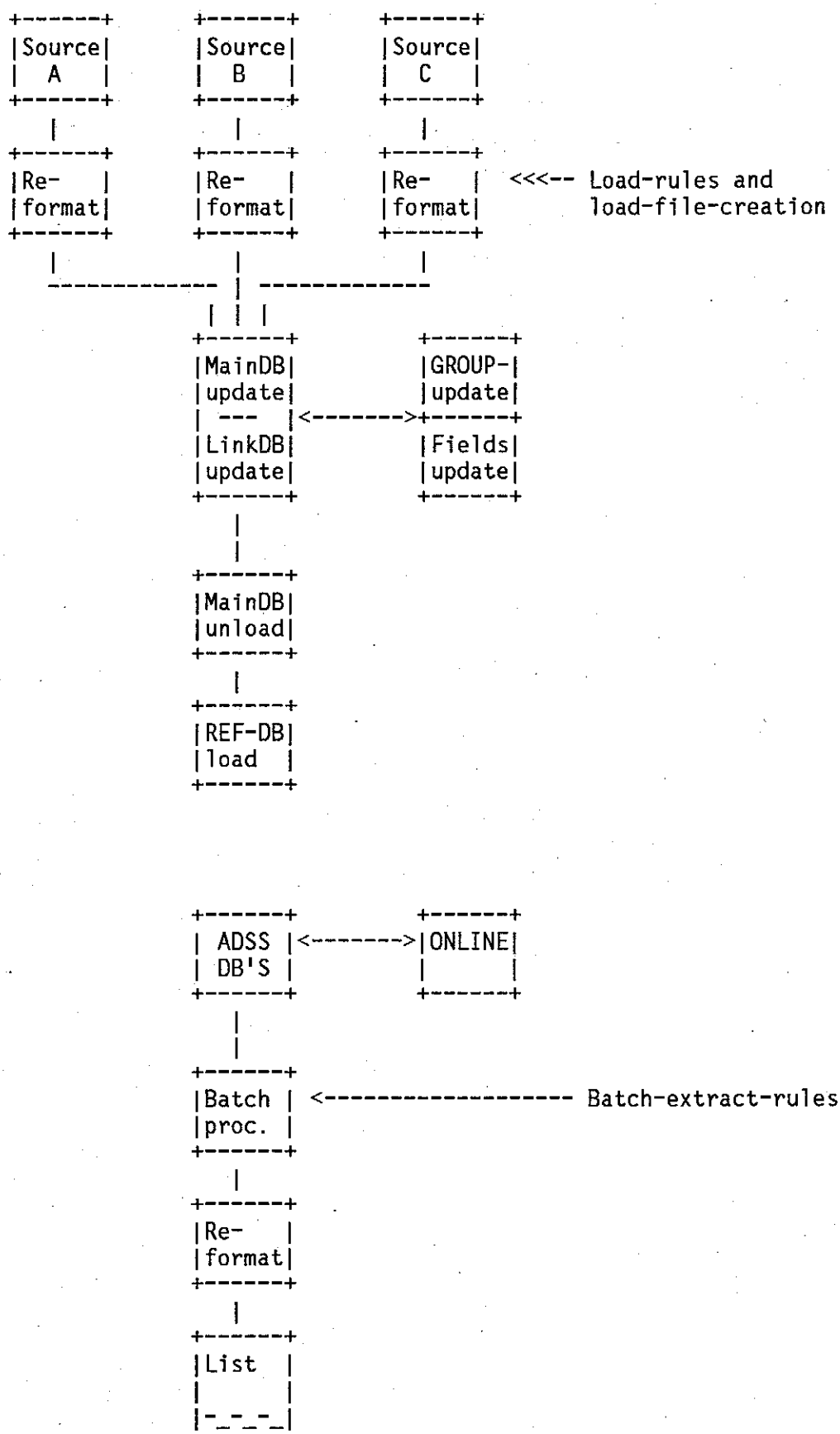
7. Make now a box-drawing-chart by each field you need and simulat at least three batch-load-days. See GG.

8. Fill out a paper-screen and simulate the future access-rules.

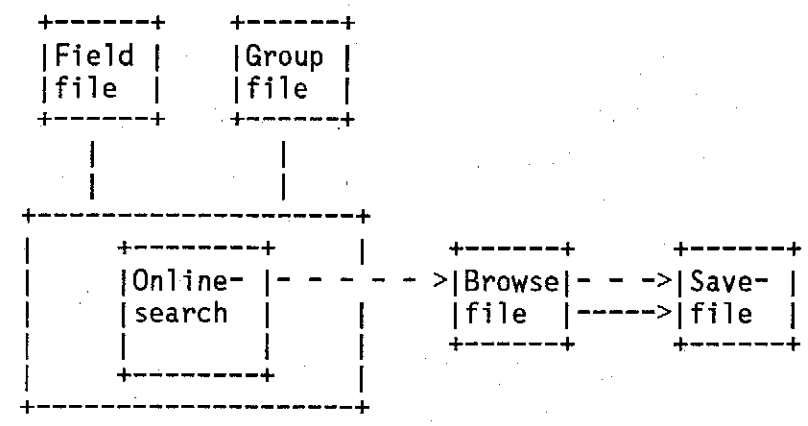
9. Is it possible to do that repetitivly? If "NO" go to point 5 again.

10. Now you have found the access-rules to run it in batch-mode. Make an application-description, including the results of point 5, 6 and 7.

KK. Load/Retrieve-processing



LL. Online-program structure.



MM. Anhang

Within the rolling link-list the backwards-links will be first lost, if the maximum of 4031 elements is exeeded.
Removed links will be saved in a lost-link-dataset.

The system is able to manage 8'388'608 different values, if the requested space is awailable (1600 CYL-3380).

The maximum space the system can occupy is:

Main-DB	840 Cyl 3380	-	8'388'608 values stored
Reference-DB	700 Cyl 3380		
Fieldref-File	1 Cyl 3380		
Group-FIle	1 Cyl 3380		
Link-DB	15 Packs 3380/Mod E	-	15% of all possible links.

A practicle size of the system will be:

Main-DB	120 Cyl 3380	-	1'200'000 values stored
Reference-DB	100 Cyl 3380		
Fieldref-File	1 Cyl 3380		
Group-FIle	1 Cyl 3380		
Link-DB	300 Cyl 3380	-	for a common application
Link-DB	400 Cyl 3380	-	for security-purpose application

Batch Jobs

Load input.
Reorg DB's.
Delete unused Main- and Ref-DB-Records.
Remove backward-links, if value has changed to constant-value.
Delete items by date (clean link-db's)

EOM

ADSSADD

```
 /*ADSSADD  I ADSS  - ADD ARG AND FUN TO ARRAY       *                  */
 /*         -861014 STALDER H.U. ISO  IS     BATCH  PLI ISD             */
 /*         -                                                           */
 /*********************************************************************/
 /*  00.00.00.00.  LOAD ARGUMENT AND FUNCTION INTO ARRAY (FUNCLOAD)   */
 /*********************************************************************/
 ADSSADD:
   PROC(PTR,ARGUM,ARGNUM,ARGLEN,ARGCNT,FUNCT, FUNLEN);
 - DCL
     MODNAME CHAR(8) INIT('ADSSADD');
 - DCL
     ARGUM CHAR(*),
     FUNCT CHAR(*);
 - DCL
     (ARGCNT,
     ARGNUM,
     ARGLEN,
     FUNLEN ) BIN FIXED(31);
 - DCL
     PTR POINTER;
 - DCL
     1 ARR BASED(PTR),
       2 HIGHNO BIN FIXED(31),
       2 ALEN BIN FIXED(31),
       2 FLEN BIN FIXED(31),
       2 DATA(ARGCNT REFER (HIGHNO)),
         3 ARG CHAR(ARGLEN REFER (ALEN)),
         3 FUN CHAR(FUNLEN REFER (FLEN));
-/*********************************************************************/
 /*  ON-UNITS                                                         */
 /*********************************************************************/
 - ON ERROR SNAP
     BEGIN;
       ON ERROR SYSTEM;
       PUT SKIP EDIT
         ('ADSSADD',ARGUM,ARGNUM,ARGLEN,ARGCNT,FUNCT,FUNLEN)
         (A,A,F(7),F(7),F(7),A,F(7));
     END;
-/*********************************************************************/
 /*  INIT                                                             */
 /*********************************************************************/
 -   ARR.HIGHNO = ARGCNT + 1;
     ARR.ALEN = ARGLEN;
     ARR.FLEN = FUNLEN;
 -   IF ARGCNT = 0 THEN
       DO;
-/*********************************************************************/
 /*   INSERT FIRST ELEMENT                                            */
 /*********************************************************************/
         ARR(1).ARG = SUBSTR(ARGUM,1,ARGLEN);
         IF FUNLEN > 0 THEN
           ARR(1).FUN = SUBSTR(FUNCT,1);
       END;
 -   ELSE
 /*********************************************************************/
 /*   IF ELEMENT DOES NOT YET EXIST, INSERT IN ITS ACENDING POSITION  */
 /*********************************************************************/
       DO;
         IF ARGNUM > 1 THEN
           L = ARGNUM + 1;
         ELSE
           L = 1;
         N = ARGCNT;
         DO J = N TO L BY -1;
           DATA.ARG(J+1) = DATA.ARG(J);
           IF FUNLEN > 0 THEN
             DATA.FUN(J+1) = DATA.FUN(J);
         END;
         DATA.ARG(L) = SUBSTR(ARGUM,1,ARGLEN);
         IF FUNLEN > 0 THEN
           DATA.FUN(L) = SUBSTR(FUNCT,1,FUNLEN);
       END;
 - END ADSSADD;
```

ADSSFDB

```
 /*ADSSFDB  I ADSS - FIELD-SEARCH-DB STRUCTURE       *                  */
 /*         -861015 STALDER H.U. ISO ADSS    BATCH  PLI PROD            */
 /*         -861015 STALDER H.U. TM    INITIAL LAYOUT                   */
 /*         -                                                           */
    2 KEY,                          /* LENGTH 6 BYTE                  */
      3 FIELDNO      BIN FIXED(15) UNALIGNED /*                        */,
      3 TESTIND      BIN FIXED(15) UNALIGNED /*                        */,
      3 RECCNT       BIN FIXED(15) UNALIGNED /*                        */,
    2 DATA,                         /*                                */
      3 FDBMAX       BIN FIXED(31) UNALIGNED /*                        */,
      3 FDBALEN      BIN FIXED(31) UNALIGNED /*                        */,
      3 FDBFLEN      BIN FIXED(31) UNALIGNED /*                        */,
      3 FDBTAB(10916),              /*                                */
        4 RECNO BIT (24) UNALIGNED      /* 3 BYTE LENGTH               */
```

ADSSFLD

```
 /*ADSSFLD  I ADSS - FIELD-TABLE                    *                  */
 /*         -861013 STALDER H.U. ISO ADSS    BATCH  PLI PROD            */
 /*         -861013 STALDER H.U. TM    INITIAL LAYOUT                   */
 /*         -                                                           */
    2 FIELDNO  PIC'ZZZZ'       /*                                      */,
    2 FIELDNAM CHAR(8)         /*                                      */,
    2 TRUENAME CHAR(8)         /*                                      */,
    2 SHORTNAM CHAR(8)         /*                                      */,
    2 DESCR    CHAR(30)        /*                                      */
```

ADSSGRP

```
 /*ADSSGRP  I ADSS - GROUP/FIELDS REFERENCE-STRUCT.  *                  */
 /*         -861015 STALDER H.U. ISO ADSS    BATCH  PLI PROD            */
 /*         -861015 STALDER H.U. TM    INITIAL LAYOUT                   */
 /*         -                                                           */
    2 KEY,                          /* LENGTH  4 BYTE                 */
      3 GROUPNO      PIC'ZZZZ'     /*                                  */,
    2 DATA,                         /*                                */
      3 GROUP        CHAR(8)       /*                                  */,
      3 UPTEDATE     CHAR(8)       /*                                  */,
      3 FIELDCNT     PIC'ZZ9'      /*                                  */,
      3 FILL         CHAR(1)       /*                                  */,
      3 FIELDTAB(110),             /*                                  */
        4 FIELDNO    PIC'Z9'       /*                                  */
```

ADSSINP

```
/*ADSSINP  I ADSS - INPUT LOAD STRUCTURE                *
/*        -861013 STALDER H.U. ISO ADSS    BATCH  PLI PROD
/*        -861013 STALDER H.U. TM    INITIAL LAYOUT
/*        -
  2 KEY,                      /* LENGTH 18 BYTE
    3 DATE          CHAR(6)   /*
    3 TIME          CHAR(4)   /*
    3 GROUP         CHAR(8)   /*
  2 DATA,                     /*
    3 REGIND        CHAR(1)   /*
    3 FLDNAME       CHAR(8)   /*
    3 LCOUNT        PIC'Z'    /* LINE-COUNT NUMERIC 0-9 (TEXT ONLY)
    3 OPCODE        CHAR(3)   /*
    3 RETAIN        PIC'ZZZZ' /*
    3 VALUE         CHAR(45)  /*
```

ADSSLOG

```
/*ADSSLOG  I ADSS - LOG STRUCTURE (SEQU. UPDATE)    *
/*        -861110 STALDER H.U. ISO ADSS    BATCH  PLI PROD
/*        -861110 STALDER H.U. TM    INITIAL LAYOUT
/*        -
  2 KEY,                      /* LENGTH 40 BYTE
    3 APPLIC        CHAR(20)  /*
    3 FILI          CHAR(1)   /*
    3 TESTNO        PIC'Z9'   /*
    3 FIL2          CHAR(1)   /*
    3 STARTED       CHAR(15)  /*
    3 FIL3          CHAR(1)   /*
  2 DATA,                     /*
    3 ENDED         CHAR(15)  /*
    3 FIL4          CHAR(1)   /*
    3 STARTNO PIC'ZZZZZZ9'    /*
    3 FIL5          CHAR(1)   /*
    3 STOPNO  PIC'ZZZZZZ9'    /*
    3 FIL6          CHAR(1)   /*
    3 RUNTIME PIC'ZZZZZZ9'    /*
```

ADSSLDB

```
/*ADSSLDB  I ADSS - LINK-DB - VALUE STRUCTURE       *
/*        -861015 STALDER H.U. ISO ADSS    BATCH  PLI PROD
/*        -861015 STALDER H.U. TM    INITIAL LAYOUT
/*        -
  2 KEY,                      /* LENGTH 8 BYTE
    3 RECNO         BIN FIXED(31) UNALIGNED /*
    3 TESTIND       BIN FIXED(15) UNALIGNED /*
    3 RECCNT        BIN FIXED(15) UNALIGNED /*
  2 RESEVE          CHAR(4)                 /*
  2 ELE,                      /*
    4 GROUPNO       PIC'ZZZZ'     UNALIGNED /*
    4 LIMIT         BIN FIXED(31) UNALIGNED /*
    3 DATA,                     /*
      4 LINKCNT     BIN FIXED(31) UNALIGNED /*
      4 LNKALEN     BIN FIXED(31) UNALIGNED /*
      4 LNKFLEN     BIN FIXED(31) UNALIGNED /*
      4 LINKTAB(3272),          /*
        5 DSTAMP    BIT(32) UNALIGNED       /*
        5 FIELDNO   BIT(16) UNALIGNED       /*
        5 STATUS    BIT(01) UNALIGNED       /* 1 = DELETED, 0 = ACTIVE
        5 TEXTDB    BIT(01) UNALIGNED       /* 1 = TDB,     0 = MDB
        5 DIRECT    BIT(01) UNALIGNED       /* 0 = RECNO,   1 = DIRECT
        5 NUMERIC   BIT(01) UNALIGNED       /* 1 = NUMERIC, 0 = CHAR
        5 RESERVE   BIT(04) UNALIGNED       /* FUTURE USE
        5 VALUE,                            /* USED FOR DIRECT-VALUE
          6 RECNO   BIT(24) UNALIGNED       /* 1TH RECNO IS IT-SELF
```

ADSSMDB

```
/*ADSSMDB  I ADSS - MAIN-DB    STRUCTURE           *
/*        -861015 STALDER H.U. ISO ADSS    BATCH  PLI PROD
/*        -861015 STALDER H.U. TM    INITIAL LAYOUT
/*        -
  2 DATA,                     /*
    3 RECNO BIN FIXED(31) UNALIGNED /*
  2 KEY,                      /* LENGTH 47 BYTE
    3 VALUE         CHAR(45) UNALIGNED /*
    3 FIELDNO       BIN FIXED(15) UNALIGNED /*
```

ADSSRDB

```
 /*ADSSRDB  I ADSS - RECNO/VALUE REFERENCE-DB STRUCT *                  */
 /*        -861015 STALDER H.U. ISO ADSS    BATCH  PLI PROD           */
 /*        -861015 STALDER H.U. TM    INITIAL LAYOUT                  */
 /*        -                                                          */
   2 KEY,                          /* LENGTH  4 BYTE                 */
     3 RECNO BIN FIXED(31) UNALIGNED /*                               */,
   2 DATA,                         /*                                 */
     3 VALUE CHAR(47)              /* DATA (45) VAR || FIELDNO (2)    */
```

ADSSSRC

```
 /*ADSSSRC  I ADSS  - FIND POS WITHIN ARRAY(BINSRCH) *                 */
 /*        -861014 STALDER H.U. ISO  IS     BATCH  PLI ISD            */
 /*        -                                                          */
 /************************************************************************/
 /*  00.00.00.00.  FIND POSITION OF ARGUM WITHIN ARRAY              */
 /************************************************************************/
 ADSSSRC:
  PROC(PTR,ARGUM,ARGNO,ARGLEN,UPPERL,FUNLEN,ELFND);
- DCL
    MODNAME CHAR(8) INIT('ADSSSRC');
- DCL
    ARGUM CHAR(*);
- DCL
    NULL BUILTIN;
- DCL
    (ARGLEN,
    ARGNO,
    HIGHNO,
    UPPERL,
    FUNLEN) BIN FIXED(31);
- DCL
    ELFND BIT(1);
- DCL
    PTR POINTER;
-   ELFND = '0'B;
    ARGNO = 1;
  DCL
    (LOWNO,
    SEARCHNO) BIN FIXED(31);
- DCL
    1 ARR BASED(PTR),
      2 HIGHNO BIN FIXED(31),
      2 ALEN BIN FIXED(31),
      2 FLEN BIN FIXED(31),
      2 DATA(UPPERL REFER (ARR.HIGHNO)),
        3 ARG CHAR(ARGLEN REFER (ARR.ALEN)),
        3 FUN CHAR(FUNLEN REFER (ARR.FLEN));
-/************************************************************************/
 /*  ON-UNITS                                                        */
 /************************************************************************/
- ON ERROR SNAP
     BEGIN;
       ON ERROR SYSTEM;
       PUT SKIP EDIT
           ('ADSSSRC',ARGUM,ARGNO,ARGLEN,UPPERL,FUNLEN,ELFND)
           (A,A,F(7),F(7),F(7),F(7),A);
     END;
-   IF PTR = NULL THEN
      SIGNAL ERROR;
-/************************************************************************/
 /*   IF TABLE NOT YET LOADED, RETURN                                */
 /************************************************************************/
-   IF UPPERL < 1 THEN
      RETURN;
-   IF UPPERL = 1 THEN
      DO;
        IF SUBSTR(ARGUM,1,ARGLEN) = DATA(1).ARG THEN
          DO;
            ELFND = '1'B;
            ARGNO = 1;
            RETURN;
          END;
        ELFND = '0'B;
        IF SUBSTR(ARGUM,1,ARGLEN) < DATA(1).ARG THEN
          ARGNO = 1;
        ELSE
          ARGNO = 2;
        RETURN;
      END;
-/************************************************************************/
 /*   SET UP BINARY-SEARCH VALUES                                    */
 /************************************************************************/
    ARR.HIGHNO = UPPERL;
    ARR.ALEN = ARGLEN;
    ARR.FLEN = FUNLEN;
    ELFND = '0'B;
    LOWNO = 1;
    HIGHNO = UPPERL;
-/************************************************************************/
 /*   BINARY-SEARCH PROCESS                                          */
 /************************************************************************/
    DO WHILE(LOWNO¬=HIGHNO);
      SEARCHNO = (HIGHNO+LOWNO)/2;
      IF SUBSTR(ARGUM,1,ARGLEN) <= DATA(SEARCHNO).ARG THEN
        HIGHNO = SEARCHNO;
      ELSE
        LOWNO = SEARCHNO + 1;
    END;
-/************************************************************************/
 /*   ELEMENT FOUND                                                  */
 /************************************************************************/
    IF SUBSTR(ARGUM,1,ARGLEN) = DATA(LOWNO).ARG THEN
      DO;
        ELFND = '1'B;
        ARGNO = LOWNO;
      END;
-   ELSE
 /************************************************************************/
 /*   ELEMENT NOT FOUND, SET THE POSISTION WHERE THE ELEMENT SHOULD BE*/
 /************************************************************************/
      DO;
        IF SUBSTR(ARGUM,1,ARGLEN) < DATA(LOWNO).ARG THEN
          ARGNO = LOWNO - 1;
        ELSE
          ARGNO = LOWNO;
      END;
- END ADSSSRC;
```

ADSSTDB

```
 /*ADSSTDB  I ADSS - TEXT-DB-STRUCTURE                 *              */
 /*        -861015 STALDER H.U. ISO ADSS    BATCH  PLI PROD           */
 /*        -861015 STALDER H.U. TM    INITIAL LAYOUT                  */
 /*        -                                                          */
   2 KEY,                          /* LENGTH 14 BYTE                 */
     3 FIELDNO BIN FIXED(15) UNALIGNED /* BIN FIXED POSITIVE ONLY     */,
     3 RECMAIN BIN FIXED(31) UNALIGNED /*                             */,
     3 TESTIND BIN FIXED(15) UNALIGNED /*                             */,
     3 LCOUNT  BIN FIXED(15) UNALIGNED /*                             */,
   2 DATA,                         /*                                 */
     3 RECSLAV BIN FIXED(31) UNALIGNED /* FUTURE USE                  */,
     3 DSTAMP  BIN FIXED(31) UNALIGNED /*                             */,
     3 TEXT    CHAR(150)           /* VARIABLE-LENGTH                 */
```

ADSSTRNS

```
/*ADSSTRNS I ADSS - TRANSLATE LOWER-CASE TU UPPER   *                  */
/*         -861127 STALDER H.U. ISO ADSS    BATCH  PLI PROD           */
/*         -861127 STALDER H.U. TM     INITIAL VERSION                */
/*         -                                                          */
  WORKCMD = TRANSLATE(ADSSCMD,
            'ABCDEFGHIJKLMNOPQRSTUVWXYZ',
            'abcdefghijklmnopqrstuvwxyz');
```

ADSSTXT

```
/*ADSSTXT  I ADSS - TEXT-DB-STRUCTURE                 *               */
/*         -861015 STALDER H.U. ISO ADSS    BATCH  PLI PROD           */
/*         -861015 STALDER H.U. TM     INITIAL LAYOUT                 */
/*         -                                                          */
  2 KEY,                        /* LENGTH  7 BYTE                    */
    5 FIELDNO BIN FIX(11) UNALIGNED  /* BIN FIXED POSITIVE ONLY */,
    5 DSTAMP  BIT(20) UNALIGNED      /* BIN FIXED POSITIVE ONLY */,
    3 RECNO   BIN FIX(23) UNALIGNED  /*                         */,
  2 DATA,                       /*                                   */
    3 TEXT    CHAR(45) VAR      /*                                   */,
```