Die PL/I-

E/A

Ein Beitrag zum

RHRK-Seminar über den

Vergleich von Programmiersprachen

Juni 1978

Joachim Backes
Regionales Hochschulrechenzentrum
der Universität Kaiserslautern

Inhaltsverzeichnis

1. Einführung

Durch die PL/I-E/A hat ein PL/I-Programmierer die Möglichkeit, Daten zwischen einem PL/I-Programm und den auf externen Medien lagernden Dateien auszutauschen. Die Zugriffsart wird durch Dateiattribute festgelegt.

Bevor auf eine Datei zugegriffen werden kann, muss sie per `OPEN` eröffnet werden, wobei die Attribute gegebenenfalls ergänzt und auf Konsistenz überprüft werden. Attribute, die durch `DCL FILE...` festgelegt werden, gelten für den gesamten Programmlauf; die bei `OPEN` angegebenen gelten nur bis `CLOSE` und sind danach wieder änderbar. Ergeben sich bei `OPEN` Diskrepanzen, wird die Bedingung `UNDEFINEDFILE` gesetzt, und `OPEN` wird abgelehnt.

Dateien, auf die nicht mehr zugegriffen wird, können per `CLOSE` abgeschlossen werden.

Man unterscheidet in PL/I zwischen Strom-Dateien und Datensatz-Dateien (`STREAM`, `RECORD`). Stromdateien bestehen i. A. aus lesbaren Schriftzeichen, die interpretierbar sind. Bei der Ein-/Ausgabe werden ggf. Konvertierungen vorgenommen.

Datensatz-Dateien sind eine 1-1-Abbildung der internen Darstellung im PL/I-Programm; beim Transport finden keine Konvertierungen statt, so dass die Verantwortung bezüglich Konsistenz mit der Dateibeschreibung voll beim Benutzer liegt.

Der Kontakt zwischen dem Betriebssystem und dem die Dateien bearbeitenden PL/I-Programm geschieht bei `OPEN` über das `TITLE`-Attribut; ansonsten werden die `FILE`-Konstanten als Kontakt-Bezüge verwendet. Dateien können aber auch über `FILE`-Variable angesprochen werden, am Ende einer Verweiskette muss aber immer eine Konstante stehen. Im Fehlerfall wird `ONFILE` gesetzt.

AIs Voreinstellung setzt die PL/I-E/A bei der stromorientierten Zuweisung `GET FILE(SYSIN)` ein, bei `PUT` wird `FILE(SYSPRINT)` eingesetzt. Wird dynamisch ein E/A-Statement ausgeführt und die Datei war noch nicht eröffnet, wird ein implizites `OPEN` einschließlich sämtlicher Abprüfungen gegeben (wobei wiederum die Bedingung `UNDEFINEDFILE` gesetzt sein kann!).

`RECORD`-Dateien bieten neben der unformatierten E/A zusätzlich die Möglichkeit, mit sogenannten verbundenen Puffern zu arbeiten, indem man bei der Eingabe statt eines Ziel-Bezuges per `READ...SET...` einen Pointer angibt, der dann nach dem Transport auf den (internen) Satzpuffer zeigt. (Man spart damit den Transport von internen Puffern in den Programmpuffer). Vor jeder `READ`-Anweisung wird ein evtl. vorhandener Verbund-Puffer gelöscht.

`READ...SET...` entspricht bei der Ausgabe das `LOCATE`-Statement. Dabei wird für den nächsten auszugebenden Datensatz ein Verbund-Puffer bereitgestellt (so als ob `ALLOCATE... SET...` gegeben würde). Satzlänge und -schlüsseI werden für die eigentliche Ausgabe bereitgehalten. Der Transport selbst erfolgt erst bei der nächsten

LOCATE-, WRITE- oder CLOSE-Anweisung. LOCATE und WRITE sind wechselseitig verwendbar.

Die LOCATE-Anweisung darf nur bei Dateien angewandt werden, die mit dem Attribut UPDATE eröffnet wurden.

1.1. Feste Dateiwerte

Feste Dateiwerte sind solche Dateigrößen, die bei der Eröffnung bestimmt werden und bis zum Abschluss der Datei fest bleiben. Sie existieren nur bei STREAM-Dateien.

- PAGESIZE (für STREAM OUTPUT PRINT) gibt die maximale Zeichenzahl pro Seite an; Voreinstellung PAGESIZE(∞).
- LINESIZE (STREAM OUTPUT) gibt die Anzahl der Zeichen pro Zeile an. Unter Umständen erfolgt ein implizites SKIP bei COLUMN- oder X-Format .
- TAB (STREAM OUTPUT PRINT) für PUT DATA und PUT LIST steuert die Anfangsposition einer Teilfolge; eine Veränderung ist nicht überall möglich.

1.2. Aktuelle Dateiwerte

Aktuelle Dateiwerte sind solche Dateigrößen, die sich dynamisch während des Programmlaufes ergeben.

- Page-Number (STREAM OUTPUT PRINT); wird am Anfang auf 1 gesetzt, erhöht sich bei jeder neuen Seite um 1, auch bei PAGE in der PUT-Anweisung. Die Pseudo-Funktion PAGEND( ) liefert die aktuelle Seitenummer, durch sie kann die Seitennummer auch neu gesetzt werden.
- Current Line (STREAM OUTPUT), beinhaltet die aktuelle Zeilen-Position. Bei PRINT-Dateien ist der Wert über die Pseudo-Funktion LINEND( ) lieferbar, eine Veränderung ist aber nicht möglich.
- Current Position (STREAM), ist dem Benutzer nicht zugänglich. Zeigt hinter das letzte eingegebene bzw. ausgegebene Zeichen bei GET bzw. PUT.
- Current record (aktueller Datensatz), interne Hilfsgröße bei Dateien vom Attribut RECORD. Mögliche Werte:

Anfang	Verweist auf Datei-Anfang, nur möglich bei SEQUENTIAL-Dateien
Ende	Weist bei SEQUENTAL-Dateien auf das Datei-Ende, in diesem Falle wird die ENDFILE- Bedingung gesetzt.
Datensatz	Verweist auf den aktuellen Satz in der Datei.
undefiniert	Nur bei Dateien vom Attribut DIRECT möglich, weist auf einen undefinierten Datensatz (Datei neu eröffnet, Satz wurde gelöscht, oder KEY-Bedingung gesetzt).

Beim Eröffnen einer Datei ergeben sich für den current record folgende Einstellungen:

DIRECT	undefiniert
OUTPUT,¬DIRECT	Anfang
Alle anderen Fälle	Anfang

Wird speziell mit dem Attribut SEQUENTIAL eröffnet, so ergeben sich folgende CURRENT-RECORD-Werte:

OPEN	Ende
READ	nächster Satz
READ IGNORE	der an der jeweiligen Position stehende Satz
WRITE	nächster Satz
LOCATE REWRITE	wird durch aktuellen Satz ersetzt, also unverändert
DELETE	der aktuelle Satz ist gelöscht, der Vorgänger wird zum aktuellen Satz, ggf. auch der Dateianfang.

Beim Eröffnen einer Datei mit dem Attribut DIRECT KEYED ergeben sich bei den nachstehenden Anweisungen folgende CURRENT-RECORD-Werte nach:

OPEN	undefiniert
READ REWRITE	falscher Schlüssel: unverändert Schlüssel vorhanden: Dieser Satz wird zum aktuellen Satz
DELETE	Wie REWRITE, READ; wenn der Satz gelöscht wird, undefiniert
WRITE,LOCATE	Falls der Satz eingetragen worden ist, ergibt sich der aktuelle Satz aus KEY, sonst unverändert

Wird die Datei mit SEQUENTIAL KEYED eröffnet, so ergibt sich der aktuelle Satz beim Zugriff ohne KEY wie bei SEQUENTIAL, beim Zugriff mit KEY wie bei DIRECT KEYED.

Allgemein gilt noch folgendes: Ist der aktuelle Satz undefiniert, so darf nicht mit READ, WRITE oder DELETE gearbeitet werden (bei SEQUENTIAL),da sonst die Bedingung ERROR gesetzt wird.

Folgende Werte können aus der Datei bestimmt bzw. in die Datei übergeben werden:

Bei RECORD-Dateien	Datensatz Datensatz-Länge Satz-Schlüssel
Bei STREAM-Dateien das Zeichenzahl	Zeilenmarke Seitenmarke Wagenrücklauf

,

Ein RECORD-Datensatz wird bei der E/A nicht konvertiert, mit dem Satzpuffer unverträgliche Längen werden ggf. nach einer Meldung korrigiert, abhängig von den Kenndaten der Datei und der Länge des auszugebenden Satzes; d. h. es wird verkürzt oder mit Füll-Elementen aufgefüllt (Bedingung RECORD!). Als Bezüge sind verschachtelte Matrizen und Matrix-Bezüge mit "*" nicht erlaubt.

Beim Schreiben in KEYED-Dateien muss ein Schlüssel angegeben werden, ebenfalls beim Lesen aus DIRECT KEYED-Dateien (sonst Bedingung KEY!). Bei SEQUENTIAL KEYED kann ohne Schlüsselangabe gelesen werden, wobei man sich den Schlüssel des gelesenen Satzes in eine CHAR-Variablen ausgeben lassen kann.

Bei der STREAM-Ein- und Ausgabe sind alle Code-Zeichen erlaubt, der Strom wird aus Teilfolgen von Zeichen aufgebaut (PUT, GET), die lückenlos aneinanderschließen, wobei Zeilenmarken, bei PRINT auch Seitenmarken und Wagenrückläufe beliebig eingefügt werden können; Zeilenmarken durch das SKIP-Attribut bzw. durch SKIP- bzw. LINE-Formate.

Seitenmarken sind durch das PAGE-Format explizit generierbar, implizit werden sie erzeugt beim Erreichen des Seiten-Endes (PAGESIZE!); die Bedingung ENDPAGE wird gesetzt, wenn die aktuelle Seiten-Nummer sich ergibt aus maximale+1; dadurch ist ein Abschalten des Seitenvorschubes möglich, wenn ENDPAGE ignoriert wird.

Wagenrückläufe werden erzeugt durch PUT SKIP(0) oder mit dem SKIP-Format, Anzahl = 0. Ein Übereinanderdrucken von Zeichen ist damit möglich.

2. Beschreibung der einzelnen E/A-Anweisungen

2.1.OPEN

```
OPEN {{FILE(Bezug)[Datei] [ENV[IRONMENT](Angabe)] [TITLE(Ausdruck)]},...;

Datei ::= { [STREAM][{INPUT|[OUTPUT][PRINT]}]
          { [STREAM][OUTPUT][PRINT][PAGESIZE(Ausdruck)][LINESIZE(Ausdruck)]
          { [RECORD][ INPUT  ] {[SEQUENTIAL]|[SEQL]}          [KEYED]]
          {          OUTPUT  } [DIRECT]
          {          UPDATE  }
```

Aufgaben der `OPEN`-Anweisung:

- Attribute vervollständigen
- Attribute prüfen
- Dateikenndaten bzgl. Attribute prüfen
- Zustand *eröffnet* erzeugen

Werden bei `OPEN` keine expIiziten Angaben gemacht, so gilt `STREAM INPUT SEQUENTIAL` aIs Voreinstellung, lautet der Dateiname `SYSPRINT`, so wird `STREAM OUTPUT` ergänzt. Wird die Datei mit `STREAM` eröffnet, werden ggf. die Größen `PAGESIZE`, `LINESIZE` und `TAB` ergänzt.

vorhanden	ergänzen
DIRECT	RECORD KEYED
KEYED	RECORD
PRINT	STREAM OUTPUT
SEQUENTIAL	RECORD
UPDATE	RECORD

Die nachstehende Tabelle führt die Systemvoreinstellung und möglichen Alternativen bei Datei-Attributen auf:

Alternativen	System-Vorgabe
STREAM | RECORD	STREAM
INPUT | OUTPUT | UPDATE	INPUT
SEQUENTIAL | DIRECT	SEQUENTIAL
vorhanden	**ergänzen**
SYSPRINT EXTERNAL STREAM OUTPUT	PRINT
vorhanden	**ergänzen**
STREAM OUTPUT	LINESIZE (x)
PRINT	PAGESIZE(∞)
	TAB(1,11,21,31,...)

x: max. Zeilenlänge des Gerätes; wenn unbekannt: 120

Wird auf eine Datei erstmalig zugegriffen ohne explizites `OPEN`, so wird intern ein `OPEN` gegeben gemäß der nachstehenden Tabelle

Anweisung	implizites OPEN mit
`GET`	`STREAM INPUT`
`PUT`	`STREAM OUTPUT`
`READ`	`RECORD [INPUT]`
`WRITE`	`RECORD [OUTPUT]`
`REWRITE`	`RECORD UPDATE`
`LOCATE`	`RECORD OUTPUT`
`DELETE`	`RECORD UPDATE`
	`[...]` wenn nicht bereits `UPDATE`

2.2.CLOSE

Aufbau: `CLOSE FILE(`Bezug`)`,...

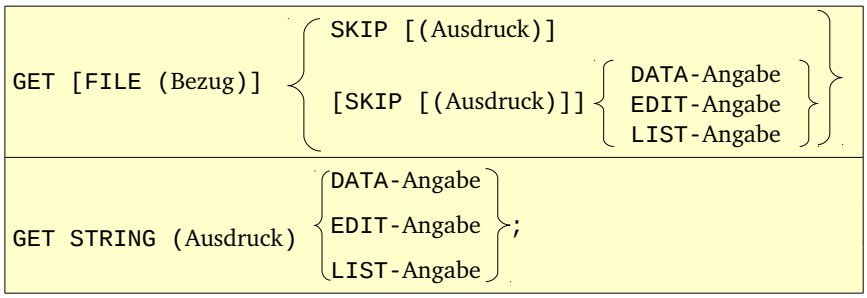
Die Datei wird abgeschlossen; alle durch `OPEN` definierten Größen werden auf undefiniert gesetzt außer denen, die bereits im `DECLARE`-Statement vorhanden waren. War die letzte Anweisung ein `LOCATE`, wird der Puffer aus- und danach freigegeben.

War die letzte Anweisung `PUT`, so wird ein `PUT SKIP` eingefügt.

Implizit werden bei Programmende alle eröffneten Dateien mit `CLOSE` geschlossen.

2.3.GET (Strom-Eingabe)

Die GET-Anweisung ist nur zulässig, wenn die Datei mit STREAM INPUT eröffnet worden ist.



Aus dem Eingabestrom werden Teilfolgen ausgeblendet, die durch DATA, EDIT oder LIST definiert sind.

DATA	Die Teilfolge hat die Form "variable=Literal-Konstante {,| }", wobei als Trenn-zeichen Komma oder Blank zählen. Eine GET-Anweisung wird beendet durch ein Semikolon im Eingabestrom.
EDIT	Die Anzahl der Teilfolgen ergibt sich aus der Anzahl der Ziel-Variablen, ihre Längen sind jeweils durch die Format-Angaben festgelegt.
LIST	Die Anzahl der Teilfolgen ergibt sich aus der Anzahl der Zielvariablen; als Trennzeichen gelten Blank und Komma, bei leeren Teilfolgen ( ) bleibt der Wert der Variablen unverändert.

Anstatt aus einer Datei kann auch ein String Eingabemedium sein. Dann ist SKIP(. . .) naturgemäß nicht erlaubt.

Beispiele:

Statement	Eingabedaten
`GET DATA ()`	`...z=w[{,| } z=w] ...`
`GET DATA (Z)`	
`GET EDIT (z,...) (f,...)`	`a ...`
`GET LIST (z,...)`	`... w [{, } w] ...`

2.3.1 DATA

Bei `DATA` sind folgende Variablenbezüge zugelassen:

- Einfacher Bezug
- Matrix-Bezug
- Struktur-Bezug
- Zeiger-Bezug

Bei falschen Bezügen wird die Bedingung `NAME` gesetzt; ist die Konstante im Eingabestrom unzulässig, wird `CONVERSION` gesetzt. Die Zustände sind dann über die Pseudo-`ON`-Variablen abfragbar.

2.3.2 EDIT

Bei der `EDIT`-Eingabe gibt es keine Trennzeichen, die Aufteilung wird durch das Format bestimmt. Unter (z, ...) können mit `DO`... auch Eingabeschleifen aufgeführt werden. Nicht skalare Bezüge werden in solche aufgelöst. Steuer-Formate `X`, `SKIP` und `COLUMN`, bei `GET STRING` nur `X`, erlauben Positionierungen. Format-Angaben können Wiederholungsfaktoren enthalten. Reicht bei der Ziel-Liste die Format-Menge nicht aus, wird wieder von vorne begonnen, überschüssige Formate dagegen werden ignoriert. Bei Verweis- (`R`-)Formaten wird entsprechend aufgebrochen. Bei der Konvertierung in das interne Format kann die Bedingung `CONVERSION` auftreten. In den nachstehenden Tabellen sind die möglichen Formate aufgelistet.

Datenformate:

Bedeutung	Format
Zeichenfolge	A [(Länge)]
Bitfolge	B [(Länge)]
	F (Länge)
Festpunkt	F (Länge,Punkt)
	F (Länge, Punkt,Skalierung)
	E (Länge)
Gleitpunkt	E (Länge, Punkt)
	E (Länge, Punkt, Mantisse)
	C(Festpunkt / Gleitpunkt / Maske) (,Festpunkt / ,Gleitpunkt / ,Maske)
Komplex	
Maske	P 'Maske'

Steuer-Formate:

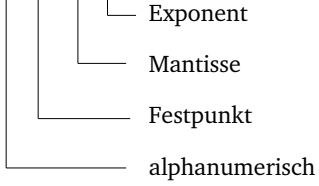
Bedeutung		Format	Anweisung		Steuer-Format			
			GET	PUT	INPUT	OUTPUT	PRINT	STRING
Seitenvorschub		PAGE	■		■	■		■
Zeilenvorschub	relativ	SKIP[(Zeilen)]						■
	absolut	LINE(Zeilennummer)	■		■	■		■
Zeichenvorsch.	relativ	X (Zeichen)	■	■				
	absolut	COL[NUM](Spalte)	■	■				■
			■	≙ Nicht erlaubt				

Verweis-Format

Bedeutung	Format
Verweis auf Format-Variable oder Format-Konstante	R (Bezug)

P-Maske:

Zeichen	Funktion	Stellen	Datentyp	Besonderheiten		
X	alphanumerische Stelle	1	A	mindestens ein X oder A, nur Kombinationen von X,A,9		
A	alphabetische Stelle					A,...Z,a,...z,SP
9	numerische Stelle					0,...,9,SP
9	Ziffernstelle	1	F M E	nicht links von Z oder *		
V	Dezimalpunkt-Angabe	-	F M E	nur einmal		
F(n)	Skalierfaktor	-	F	ohne Angabe ganz rechts		
Z	Ersetzung führender Nullen durch Leerzeichen	1	F M E	nicht rechts von 9Y	nur alternativ	ersetzt 9
*	Ersetzung führender Nullen durch *					
Y	Ersetzung einer Null durch Leerzeichen			nicht links von Z oder *		
S	Vorzeichen	1	F M E	nur einmal; nur alternativ; nicht mit CR DB		
+	Plus-Vorzeichen					
-	Minus-Vorzeichen			links oder rechts bei F		
SS...	gleitendes Vorzeichen	n	F M	nur alternativ, ersetzt n-1 mal 9;nur links von 9Y; nicht mit z*; rechts von V kein 9Y		
++...	gleitendes Plus-Vorzeichen					
--...	gleitendes Minus-Vorzeichen					
$	Währungssymbol	1	F	rechts oder links		
$$...	gleitendes Währungssymbol	n		nur links von 9Y; nicht mit Z*; ersetzt n-1 mal 9; rechts von V kein 9Y		
CR	Kreditzeichen (-)	2		nur rechts von Ziffernstellen und $; nicht mit $+-; nur einmal		
DB	Debetzeichen (-)	2				
E	Exponentenzeichen	1	M E	alternativ; nur einmal		
K	Mantisse-Exponent-Trennung	-				
,	Komma	1	A F M E	an beliebiger Stelle in beliebiger Kombination; werden in Verbindung mit Unterdrückung führender Nullen ebenfalls unterdrückt.		
.	Punkt					
/	Schrägstrich					
B	Leerzeichen					



Bei der Eingabe mit **EDIT** sind folgende Vorschriften bezüglich Format- und Teilfolgenangaben einzuhalten:

<table>
<tr><th colspan="3">Format</th><th colspan="2">ausgeblendete Teilfolge</th></tr>
<tr><td colspan="3">A(n)</td><td colspan="2" rowspan="3">n Zeichen</td></tr>
<tr><td colspan="3">B(n)</td></tr>
<tr><td>P 'Maske'</td><td colspan="2">n: alle Maskenzeichen ohne V,F,K</td></tr>
<tr><td>F(n)</td><td>≡ F(n,0,0)</td><td rowspan="3">n Zeichen
dezimale Festpunkt-Konstante
Punkt aus Teilfolge; wenn nicht vorhanden: Punkt d Stellen von rechts</td><td rowspan="3">Teilfolge: 10^s</td></tr>
<tr><td>F(n,d)</td><td>≡ F(n,d,0)</td></tr>
<tr><td colspan="2">F(n,d,s)</td></tr>
<tr><td>E(n)</td><td>≡ E(n,0)</td><td colspan="2" rowspan="3">n Zeichen
dezimale Gleitpunkt-Konstante
Exponent zur Basis 10; E oder Vorzeichen des Exponenten erforderlich;
Punkt aus Teilfolge; wenn nicht vorhanden: Punkt d Stellen von rechts</td></tr>
<tr><td colspan="2">E(n,d)</td></tr>
<tr><td>E(n,d,m)</td><td>m bedeutungslos</td></tr>
<tr><td colspan="3">C(Real- und Imag-Format)</td><td colspan="2" rowspan="2">2 Teilfolgen:
Realteil und Imaginärteil</td></tr>
<tr><td colspan="3">C(Real-Format, Imag-Format)</td></tr>
</table>

2.3.3 LIST

Bei der LIST-Eingabe wird je Variable eine Teilfolge ausgeblendet, die durch Trennzeichen (Komma oder Blank, außer in Strings) abgeschlossen wird. Bei der Konvertierung kann die Bedingung CONVERSION auftreten. Wie bei EDIT sind unter z auch DO-Schleifen angebbar. Struktur- oder Matrix-Bezüge werden in skalare Bezüge aufgelöst.

2.4.PUT (Strom-Ausgabe)

<table>
<tr><td>PUT [FILE(Bezug)][SKIP [(Ausdruck)]
[PAGE](LINE(Ausdruck))][{ DATA-Angabe
EDIT-Angabe
LIST-Angabe }]</td></tr>
<tr><td>PUT STRING (Bezug) { DATA-Angabe
EDIT-Angabe
LIST-Angabe }</td></tr>
</table>

Auf die Strom-Ausgabe wird nicht näher eingegangen, da das für die Eingabe Gesagte zum größten Teil auch für die Ausgabe gilt (nicht prinzipiell neues!).

2.5.E/A-Anweisungen bei RECORD-Attribut

Die kleinste Einheit, auf die in `RECORD` vereinbarten Dateien zugegriffen werden kann, ist der Satz. Mögliche Zugriffsarten sind `SEQUENTIAL` (`SEQL`), `SEQL KEYED` und `DIRECT KEYED`. Im letzten Fall muss bei jedem Transport ein Schlüssel angegeben werden. Die mögliche Transportrichtung wird bei `OPEN` festgelegt.

Die nachstehende Tabelle gibt darüber Auskunft, welche Anweisungen bei welchen Attribut-Sätzen erlaubt sind.

Attribut-Satz				Erlaubte Anweisung
RECORD	INPUT	SEQUENTIAL		READ [IGNORE]
			KEYED	READ [KEY[TO]] [IGNORE]
		DIRECT	KEYED	READ KEY
	OUTPUT	SEQUENTIAL		WRITE LOCATE
			KEYED	WRITE KEYFROM LOCATE KEYFROM
		DIRECT	KEYED	WRITE KEYFROM LOCATE KEYFROM
	UPDATE	SEQUENTIAL		READ [IGNORE] REWRITE DELETE
			KEYED	READ [KEY[TO]] [IGNORE] REWRITE[KEY] DELETE [KEY]
		DIRECT	KEYED	READ KEY
				WRITE KEYFROM
				REWRITE KEY
				DELETE KEY

Umgekehrt sind nur gewisse Dateiattribute, abhängig von der anzuführenden EA-Anweisung, erlaubt. Die Zusammenhänge sind aus der nachstehenden Tabelle ersichtlich.

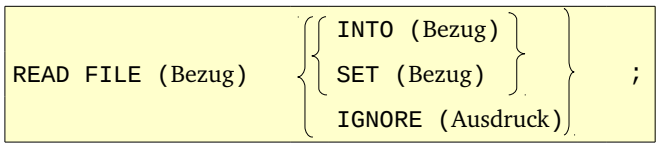
Anweisung		erlaubte Attribut-Sätze			
DELETE		RECORD	UPDATE	SEQ	[KEYED]
	KEY	RECORD	UPDATE	{SEQUENTIAL|DIRECT}	KEYED
LOCATE		RECORD	OUTPUT	SEQUENTAL	
	KEYFROM	RECORD	OUTPUT	{SEQUENTIAL|DIRECT}	KEYED
READ	[IGNORE]	RECORD	{INPUT|UPDATE}	SEQUENTIAL	[KEYED]
	KEYTO	RECORD	{INPUT|UPDATE}	SEQUENTIAL	KEYED
	KEY	RECORD	{INPUT|UPDATE}	{SEQUENTIAL|DIRECT}	KEYED
REWRITE		RECORD	UPATE	SEQUENTIAL	[KEYED]
	KEY	RECORD	UPDATE	{SEQUENTIAL|DIRECT}	KEYED
WRITE		RECORD	OUTPUT	SEQUENTIAL	
	KEYFROM	RECORD	OUTPUT	{SEQUENTIAL|DIRECT}	KEYED
		RECORD	UPDATE	DIRECT	KEYED

2.5.1 Attribut SEQL

Bei Eröffnen mit `INPUT` muss in der Datei eine Satz-Reihenfolge definiert sein, bei `OUTPUT` werden die Sätze durch dynamisch aufeinanderfolgende `WRITE`'s oder `LOCATE`'s hintereinander abgelegt, bei `UPDATE` muss eine Reihenfolge bereits definiert sein.

2.5.1.1 READ

Aufbau:



Beim (impliziten) OPEN ergibt sich der current record als Dateianfang, nach Lesen des letzten Satzes ist die Bedingung ENDFILE gesetzt. Bei falscher Satzlänge wird die RECORD-Bedingung gesetzt. Ein etwaiger, vorher verbundener Puffer, wird vor Ausführung der READ-Anweisung gesetzt.

2.5.1.2 WRITE, LOCATE

Aufbau:

```
WRITE FILE (Bezug) FROM (Bezug)
```

Eine falsche Pufferlänge erzeugt die RECORD-Bedingung.

```
LOCATE Bezeichner (Bezug) [ SET (Bezug) ]
```

Durch LOCATE wird zunächst ein Pufferspeicher angelegt. Die eigentliche Ausgabe erfolgt entweder bei der nächsten WRITE-LOCATE-Anweisung oder bei CLOSE, erst dann kann auch ggf. die Bedingung RECORD auftreten.

Wenn die Datei zusätzlich mit dem UPDATE-Attribut vereinbart wird, ist es möglich, mit REWRITE Sätze zu überschreiben bzw. mit DELETE zu löschen:

```
REWRITE FILE (Bezug) [ FROM (Bezug) ]
DELETE FILE (Bezug)
```

Falls bei REWRITE FROM nicht angegeben ist, muss der zu überschreibende Satz einen verbundenen Puffer besitzen. Der zu überschreibende Satz muss vorhanden sein. (Sonst wird die Bedingung ERROR gesetzt). RECORD kann auch auftreten.

2.5.2 Attribut DIRECT KEYED

In diesem Fall muss in jeder Transport-Anweisung ein Satzschlüssel mit angegeben werden, der den Datensatz adressiert. Der Schlüssel ist eine Zeichenfolge, andere Datentypen werden gemäß den Vorschriften nach CHAR gewandelt.

READ FILE (Bezug)	{{INFO|SET}(Bezug)}}	KEY (Ausdruck)
WRITE FILE (Bezug)	FROM (Bezug)	KEYFROM (Ausdruck)
LOCATE Bezeichner	FILE (Bezug)	[SET (Bezug)] KEYFROM (Ausdruck)

Bei unzulässigem Schlüssel wird die Bedingung KEY gesetzt. Wenn die Datei zusätzlich mit UPDATE eröffnet wurde, ist Fortschreiben bzw. Löschen mit REWRITE bzw. DELETE möglich.

REWRITE FILE (Bezug)	KEY (Ausdruck)	[FROM (Bezug)]
DELETE FILE (Bezug)	KEY (Ausdruck)	

2.5.3 Attribut SEQL KEYED

Durch das Attribut SEQL KEYED ergibt sich eine Kombination von 2.5.l und 2.5.2, ein entsprechender Mischbetrieb ist möglich, z. B.

Sequentielle Verarbeitung		
READ FILE (Bezug)	{INTO|SET}(Bezug) IGNORE(Ausdruck)	KEYTO (Bezug)
Direkte Verarbeitung		
READ FILE (Bezug) {	{{INTO|SET}(Bezug)}	KEY(Ausdruck)

WRITE und LOCATE sind ebenfalls möglich, zusätzlich lässt das Attribut UPDATE ein Fortschreiben bzw. Löschen mit REWRITE bzw. DELETE zu.