Mainframe Internet Integration: VSAM

Universität Tübingen
Mathematisch-Naturwissenschaftliche Fakultät
Fachbereich Informatik, Abteilung Technische Informatik
Prof. Dr. Wolfang Rosenstiel
Gerald Kreißig
Dr. Jens Müller

17. November 2016

Inhaltsverzeichnis

1 Vorgehen

In dieser Anleitung wird eine Einführung in das Anlegen und Benutzen von VSAM-Datasets, genauer *Key Sequenced Datasets* (KSDS), gegeben.

Das Vorgehen wird anhand eines Beispiels gezeigt:

Es sollen für die Universität Matrikelnummer, studserv-Kürzel sowie Vor- und Nachname in einem VSAM-Dataset gespeichert werden. Dabei soll in einem Programm zum einen über die Matrikelnummer und zum anderen über das studserv-Kürzel ein Zugriff auf die Daten möglich sein.

Für diesen Zweck wird zuerst ein VSAM-Dataset angelegt, dass die Daten beinhalten soll, sowie einen *Index auf die Matrikelnummer*. Das Dataset wird danach mit den Daten der Studenten gefüllt. Auf die Daten wird über ein COBOL-Programm zugegriffen.

Damit ein Zugriff auch über das studserv-Kürzel erfolgen kann, muss anschließend das VSAM-Dataset um einen *alternativen Index* erweitert werden. In einem weiteren COBOL-Programm wird über diesen Index auf die Daten zugegriffen. Eine Übersicht sehen sie in Abbildung 1.

Name	**Verwendung**
PRAKxxx.VSAM.STUDENT	VSAM-Cluster
PRAKxxx.VSAM.STUDENT.DATA	Data Component des VSAM-Cluster
PRAKxxx.VSAM.STUDENT.INDEX	Index Component des VSAM-Cluster
PRAKxxx.VSAM.STUDENT.PATH	Pfad zum VSAM Cluster
PRAKxxx.VSAM.STUDENT.ALTINDEX	VSAM-Cluster
PRAKxxx.VSAM.STUDENT.ALTINDEX.DATA	Data Component des VSAM-Cluster
PRAKxxx.VSAM.STUDENT.ALTINDEX.INDEX	Index Component des VSAM-Cluster
PRAKxxx.VSAM.SEQDATA	Sequentieller Dataset für Daten Eingabe
PRAKxxx.VSAM.COBOL(STUD)	COBOL Programm #1
PRAKxxx.VSAM.COBOL(STUD2)	COBOL Programm #2
PRAKxxx.VSAM.CNTL(DEFCLUST)	JCL-Skript
PRAKxxx.VSAM.CNTL(REPRO)	JCL-Skript
PRAKxxx.VSAM.CNTL(COMPILE)	JC- Skript
PRAKxxx.VSAM.CNTL(RUN)	JCL-Skript
PRAKxxx.VSAM.CNTL(DEFAIX)	JCL-Skript
PRAKxxx.VSAM.CNTL(COMP2)	JCL-Skript
PRAKxxx.VSAM.CNTL(RUN2)	JCL-Skript
PRAKxxx.VSAM.LOAD(STUD)	Load Module
PRAKxxx.VSAM.LOAD(STUD2)	Load Module

Abbildung 1: Übersicht der verwendeten Datasets

1.1 Erstellen des VSAM-Dataset

VSAM wird in erster Linie für Anwendungsdaten eingesetzt. Es wird nicht für die Speicherung von Quellcode, JCL oder ausführbarer Programme benutzt und kann auch nicht ohne weiteres mit ISPF editiert oder wiedergegeben werden.

Der Begriff Virtual Storage Access Method (VSAM) bezeichnet sowohl den Dataset Typ (Organisation) als auch die Zugriffsverfahren (access methods) um unterschiedlichste Daten zu verwalten. VSAM verfügt über komplexere Funktionen als andere z/OS Access Methods und verwendet ein sehr spezifisches Vorgehen, um Daten auf dem Plattenspeicher zu speichern.

Der erste Schritt bei der Generierung einer neuen relationalen Datenbank ist die Erstellung eines Datenbankschemas. Im Falle von VSAM muss als erster Schritt ein VSAM-Cluster angelegt werden.

Die Bedeutung des Begriffs Cluster, im Kontext mit VSAM, ist hier nicht zu verwechseln mit dem Cluster-Begriff aus der Parallelverarbeitung!

Des Weiteren muss das (logische) `Record Format` definiert werden. Es wird sich hier für ein `Fixed Length Record`-Format entschieden und ein Record soll aus vier Feldern bestehen:

- Matrikelnummer mit 7 Zeichen (Bytes)
- Studserv-Kennung mit 8 Zeichen
- Nachname mit 20 Zeichen
- Vorname mit 20 Zeichen

Damit besteht ein Record aus 55 Zeichen und die Daten werden in der angegebenen Reihenfolge im Record gespeichert.

Ein neues VSAM-Dataset wird mit Hilfe der z/OS System Utility `IDCAMS` (Access Method Services) erstellt. Es wird ein neues PDS (`PRAKxxx.VSAM.CNTL`) angelegt, dass alle JCL-Skripte aufnehmen soll. Für die Erstellung des Clusters bietet sich als Name `DEFCLUST` für das Member an. Das Skript zum erstellen des Clusters sieht folgendermaßen aus:

Beim erstmaligen Ausführen kommt es sehr wahrscheinlich zu einem Rückgabewert `MAXCC=8`, da mit der ersten Anweisung ein vorheriger Cluster gelöscht werden soll, der erst später (wieder) angelegt wird. Also keine Panik!

```
//PRAKxxxD JOB (),NOTIFY=&SYSUID
//DEFCLS EXEC PGM=IDCAMS,REGION=4096K
//SYSPRINT DD SYSOUT=A
//SYSIN DD *
    DELETE PRAKxxx.VSAM.STUDENT
    DEFINE CLUSTER ( -
           NAME(PRAKxxx.VSAM.STUDENT) -
           VOL(Z8RES1) -
           RECORDSIZE(55 55) -
           RECORDS(10 10) -
           KEYS(7 0) -
           INDEXED ) -
         DATA ( -
           NAME(PRAKxxx.VSAM.STUDENT.DATA) ) -
```

```
          INDEX ( -
            NAME(PRAKxxx.VSAM.STUDENT.INDEX) )
/*
```

Die beiden Parameter bei `RECORDSIZE` geben die mittlere und die maximale Größe eines Records an, die in diesem Fall gleich groß sind. `INDEXED` legt fest, dass es sich um einen KSDS handelt. `KEYS` gibt an, dass der Schlüssel für den Zugriff 7 Zeichen lang ist und bei Offset 0 beginnt, da die Matrikelnummer sich als Erstes im Record befindet.

Ein VSAM-Index kann aus einer einzigen Ebene oder aus mehreren bestehen. Jede Stufe enthält Zeiger auf die nächst tiefere Ebene (Komponente).

Nach Ausführen des Skriptes sind zwei neue Datasets angelegt worden: `PRAKxxx.VSAM.STUDENT.INDEX` und `PRAKxxx.VSAM.STUDENT.DATA`.

Auf einem Plattenspeicher nimmt ein KSDS zwei lineare Speicherbereiche ein, sogenannte Komponenten. Es gibt zwei Arten von Komponenten, die Datenkomponente und die Indexkomponente. In der Datenkomponente sind die Records enthalten. (Andere VSAM Organisationen, zum Beispiel Entry-Sequenced-Datasets (ESDS), haben nur die Daten-Komponente.) Die Indexkomponente ist eine Sammlung von Records (index logical records), die die Keys der Records aus der Datenkomponente, sowie deren Adressen enthalten.

Um den Cluster nun mit Daten zu füllen, muss eine Quelle angelegt werden aus der diese kopiert werden sollen. Dazu dient ein sequenzielles Dataset mit einer Recordlänge von 55 Bytes. Der Inhalt könnte wie in Abbildung 2 aussehen.

Per Default werden in Fixed-Format-Records beim Speichern die Zeilennummer in die letzten 8 Spalten geschrieben. Dies lässt sich über "`NUMBER OFF`" im ISPF-Editor ändern.

```
EDIT       PRAKxxx.VSAM.SEQDATA                              Columns 00001 00055
=COLS> ----+----1----+----2----+----3----+----4----+----5----+
****** ***************************** Top of Data ******************************
000100 1111111mam14abcTest                Test
000200 1234567mam10fooEbbers              Mike
000300 1234569mam07zosKettner             John
000400 3333333mam16dddParziale            Lydia
000500 4444444mam13adsLopes Alves         Edi
000600 6289482mam10dbsEgeler              Klaus
****** **************************** Bottom of Data ****************************
```

Abbildung 2: Ansicht im ISPF-Editor

Dieses Dataset wird im `REPRO`-JCL-Skript übergeben und in den VSAM-Cluster kopiert:

```
//PRAKxxxR JOB (),NOTIFY=&SYSUID
//*
//* COPY SEQENTIAL DATASET INTO VSAM CLUSTER
//*
//DEFCLS EXEC PGM=IDCAMS,REGION=4096K
//SEQDD DD DSN=&SYSUID..VSAM.SEQDATA,DISP=SHR
```

```
//VSAMDD DD DSN=&SYSUID..VSAM.STUDENT,DISP=SHR
//SYSPRINT DD SYSOUT=A
//SYSIN DD *
  REPRO INFILE(SEQDD) -
        OUTFILE(VSAMDD)
/*
```

1.2 Datenzugriff über ein COBOL-Programm

Das folgende Programm nimmt eine Matrikelnummer entgegen und gibt den zugehörigen Record aus dem VSAM formatiert wieder.

```
␣␣␣␣␣␣␣IDENTIFICATION␣DIVISION.
␣␣␣␣␣␣␣*
␣␣␣␣␣␣␣*␣Programm␣zum␣Zugriff␣auf␣VSAM-Dataset
␣␣␣␣␣␣␣*
␣␣␣␣␣␣␣PROGRAM-ID.␣␣STUD.
␣␣␣␣␣␣␣/
␣␣␣␣␣␣␣ENVIRONMENT␣DIVISION.
␣␣␣␣␣␣␣*--------------------
␣␣␣␣␣␣␣CONFIGURATION␣SECTION.
␣␣␣␣␣␣␣SPECIAL-NAMES.
␣␣␣␣␣␣␣INPUT-OUTPUT␣SECTION.
␣␣␣␣␣␣␣FILE-CONTROL.
␣␣␣␣␣␣␣␣␣␣␣␣SELECT␣STUD-FILE
␣␣␣␣␣␣␣␣␣␣␣␣ASSIGN␣TO␣STUDDS
␣␣␣␣␣␣␣␣␣␣␣␣ORGANIZATION␣IS␣INDEXED
␣␣␣␣␣␣␣␣␣␣␣␣ACCESS␣IS␣DYNAMIC
␣␣␣␣␣␣␣␣␣␣␣␣RECORD␣KEY␣IS␣MATNR
␣␣␣␣␣␣␣␣␣␣␣␣FILE␣STATUS␣IS␣FSTAT-CODE␣VSAM-CODE.
␣␣␣␣␣␣␣␣␣␣␣␣SELECT␣MATNR-IN
␣␣␣␣␣␣␣␣␣␣␣␣ASSIGN␣TO␣SYSIN
␣␣␣␣␣␣␣␣␣␣␣␣ORGANIZATION␣IS␣SEQUENTIAL
␣␣␣␣␣␣␣␣␣␣␣␣FILE␣STATUS␣IS␣MATNR-IN-CODE.
␣␣␣␣␣␣␣DATA␣DIVISION.
␣␣␣␣␣␣␣*-------------
␣␣␣␣␣␣␣FILE␣SECTION.
␣␣␣␣␣␣␣FD␣STUD-FILE
␣␣␣␣␣␣␣␣␣␣␣␣RECORD␣CONTAINS␣55␣CHARACTERS.
␣␣␣␣␣␣␣␣␣␣01␣STUDENT-RECORD.
␣␣␣␣␣␣␣␣␣␣␣␣05␣MATNR␣␣␣␣␣␣␣PIC␣X(7).
␣␣␣␣␣␣␣␣␣␣␣␣05␣STUDSERV␣␣␣␣PIC␣X(8).
␣␣␣␣␣␣␣␣␣␣␣␣05␣NNAME␣␣␣␣␣␣␣PIC␣X(20).
```

Beachten Sie, dass der COBOL-Kompiler den Quelltext spaltenabhängig interpretiert! Die ersten sechs Spalten sind für die Zeilennummern reserviert, die siebente Spalte unter anderem für das Setzen des Kommentarindikators (*).

```
␣␣␣␣␣␣␣␣␣␣␣␣05␣VNAME␣␣␣␣␣␣␣␣PIC␣X(20).
␣␣␣␣␣␣␣␣FD␣MATNR-IN
␣␣␣␣␣␣␣␣␣␣␣␣RECORDING␣MODE␣F
␣␣␣␣␣␣␣␣␣␣␣␣BLOCK␣0␣RECORDS
␣␣␣␣␣␣␣␣␣␣␣␣RECORD␣80␣CHARACTERS
␣␣␣␣␣␣␣␣␣␣␣␣LABEL␣RECORD␣STANDARD.
␣␣␣␣␣␣␣␣␣␣01␣MATNR-RECORD␣␣␣PIC␣X(80).
␣␣␣␣␣␣␣/
␣␣␣␣␣␣␣␣WORKING-STORAGE␣SECTION.
␣␣␣␣␣␣␣␣01␣STATUS-AREA.
␣␣␣␣␣␣␣␣␣␣05␣FSTAT-CODE␣PIC␣X(2).
␣␣␣␣␣␣␣␣␣␣␣␣88␣I-O-OKAY␣␣␣VALUE␣ZEROES.
␣␣␣␣␣␣␣␣␣␣05␣MATNR-IN-CODE␣␣␣PIC␣X(2).
␣␣␣␣␣␣␣␣␣␣05␣VSAM-CODE.
␣␣␣␣␣␣␣␣␣␣␣␣10␣VSAM-R15-RETURN-CODE␣␣␣␣␣PIC␣9(2)␣COMP.
␣␣␣␣␣␣␣␣␣␣␣␣10␣VSAM-FUNCTION-CODE␣␣␣␣␣␣␣PIC␣9(1)␣COMP.
␣␣␣␣␣␣␣␣␣␣␣␣10␣VSAM-FEEDBACK-CODE␣␣␣␣␣␣␣PIC␣9(3)␣COMP.
␣␣␣␣␣␣␣␣01␣WS-STUDENT.
␣␣␣␣␣␣␣␣␣␣05␣WS-MATNR␣␣␣␣␣␣PIC␣X(7).
␣␣␣␣␣␣␣␣␣␣05␣WS-STUDSERV␣␣␣PIC␣X(8).
␣␣␣␣␣␣␣␣␣␣05␣WS-NNAME␣␣␣␣␣␣PIC␣X(20).
␣␣␣␣␣␣␣␣␣␣05␣WS-VNAME␣␣␣␣␣␣PIC␣X(20).
␣␣␣␣␣␣␣␣01␣WS-MATNR-IN-RECORD.
␣␣␣␣␣␣␣␣␣␣05␣WS-MATNR-IN␣␣␣PIC␣X(7).
␣␣␣␣␣␣␣␣␣␣05␣FILLER␣PIC␣X(73).
␣␣␣␣␣␣␣/
␣␣␣␣␣␣␣␣PROCEDURE␣DIVISION.
␣␣␣␣␣␣␣␣␣␣␣␣OPEN␣INPUT␣MATNR-IN.
␣␣␣␣␣␣␣␣␣␣␣␣READ␣MATNR-IN␣INTO␣WS-MATNR-IN-RECORD.
␣␣␣␣␣␣␣␣␣␣␣␣DISPLAY␣"SUCHE␣"␣WS-MATNR-IN.
␣␣␣␣␣␣␣␣␣␣␣␣OPEN␣INPUT␣STUD-FILE.
␣␣␣␣␣␣␣␣␣␣␣␣IF␣FSTAT-CODE␣NOT␣=␣"00"
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣DISPLAY␣"OPEN␣INPUT␣VSAM␣FILE␣FS-CODE:␣"␣FSTAT-CODE
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣PERFORM␣VSAM-CODE-DISPLAY
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣STOP␣RUN
␣␣␣␣␣␣␣␣␣␣␣␣END-IF.
␣␣␣␣␣␣␣␣␣␣␣␣MOVE␣WS-MATNR-IN␣TO␣MATNR.
␣␣␣␣␣␣␣␣␣␣␣␣READ␣STUD-FILE␣RECORD␣KEY␣IS␣MATNR.
␣␣␣␣␣␣␣␣␣␣␣␣IF␣FSTAT-CODE␣NOT␣=␣"00"␣AND␣FSTAT-CODE␣NOT␣=␣"02"
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣DISPLAY␣"READ␣STUD-FILE␣FS-CODE:␣"␣FSTAT-CODE
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣PERFORM␣VSAM-CODE-DISPLAY
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣STOP␣RUN
␣␣␣␣␣␣␣␣␣␣␣␣END-IF.
␣␣␣␣␣␣␣␣␣␣␣␣MOVE␣MATNR␣TO␣WS-MATNR.
```

```
␣␣␣␣␣␣␣␣␣␣␣␣MOVE␣STUDSERV␣TO␣WS-STUDSERV.
␣␣␣␣␣␣␣␣␣␣␣␣MOVE␣NNAME␣TO␣WS-NNAME.
␣␣␣␣␣␣␣␣␣␣␣␣MOVE␣VNAME␣TO␣WS-VNAME.
␣␣␣␣␣␣␣␣␣␣␣␣CLOSE␣STUD-FILE.
␣␣␣␣␣␣␣␣␣␣␣␣CLOSE␣MATNR-IN.
␣␣␣␣␣␣␣␣␣␣␣␣DISPLAY␣"MATNR:␣␣␣␣"␣WS-MATNR.
␣␣␣␣␣␣␣␣␣␣␣␣DISPLAY␣"STUDSERV:␣"␣WS-STUDSERV.
␣␣␣␣␣␣␣␣␣␣␣␣DISPLAY␣"NACHNAME:␣"␣WS-NNAME.
␣␣␣␣␣␣␣␣␣␣␣␣DISPLAY␣"VORNAME:␣␣"␣WS-VNAME.
␣␣␣␣␣␣␣␣␣␣␣␣STOP␣RUN.
␣␣␣␣␣␣
␣␣␣␣␣␣␣VSAM-CODE-DISPLAY.
␣␣␣␣␣␣␣␣␣␣␣␣DISPLAY␣"VSAM␣CODE␣-->"
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣"␣RETURN:␣"␣VSAM-R15-RETURN-CODE,
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣"␣COMPONENT:␣"␣VSAM-FUNCTION-CODE,
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣"␣REASON:␣"␣VSAM-FEEDBACK-CODE.
```

Angenommen das Programm wurde unter `PRAKxxx.VSAM.COBOL(STUD)` abgelegt und das PDS `PRAKxxx.VSAM.LOAD` zur Aufnahme der generierten Load-Module existiert, lässt es sich mit dem folgenden Skript `COMPILE` kompilieren:

```
//PRAKxxxC JOB (),NOTIFY=&SYSUID
//*
//* COMPILIEREN DES COBOL-PROGRAMMS ZUM VSAM-ZUGRIFF
//*
//COBCOMP EXEC IGYWCL
//COBOL.SYSIN  DD DSN=&SYSUID..VSAM.COBOL(STUD),DISP=SHR
//LKED.SYSLMOD DD DSN=&SYSUID..VSAM.LOAD,DISP=SHR
//LKED.SYSIN   DD *
 NAME STUD(R)
/*
```

Zum Ausführen ist ein weiteres JCL-Skript `RUN` nötig, dass dem COBOL-Programm als Argument die gesuchte Matrikelnummer übergibt:

```
//PRAKxxxR JOB (),NOTIFY=&SYSUID
//*
//* AUSFUEHREN DES COBOL-PROGRAMMS ZUM VSAM-ZUGRIFF
//*
//RUN      EXEC PGM=STUD
//STEPLIB  DD DSN=&SYSUID..VSAM.LOAD,DISP=SHR
//STUDDS   DD DSN=&SYSUID..VSAM.STUDENT,DISP=SHR
//SYSIN    DD *
```

```
1111111
/*
```

Der `STUDDS DD`-Eintrag stellt die Verbindung für das COBOL-Programm dar, die zeigt, wo die Daten abgelegt sind. Über `SYSIN` wird die Matrikelnummer des Records übergeben, der gesucht werden soll. (* signalisiert, dass die Eingabe direkt folgt.)

Die Ausgabe dieses Jobs befindet sich im Joblog unter dem `SYSOUT` vom Step `RUN`:

```
 SDSF OUTPUT DISPLAY PRAKxxxR JOB*****  DSID   102 LINE 1       COLUMNS 02- 81
 COMMAND INPUT ===>                                          SCROLL ===> PAGE
SUCHE 1111111
MATNR:    1111111
STUDSERV: mam14abc
NACHNAME: Test
VORNAME:  Test        00000100
****************************** BOTTOM OF DATA ******************************
```

Abbildung 3: Ansicht des Joblog im SDSF

1.3 Erweitern um alternativen Index

Um nun auch über das Studserv-Kürzel auf einen Record zugreifen zu können, benötigt man einen alternativen Index-Cluster. Bevor man über über einen alternativen Index (auch Sekundärschlüssel genannt) zugreifen kann, muss ein Pfad definiert werden mithilfe dessen man auf den ursprünglichen Basis-Cluster über die alternativen Indizes zugreifen kann.

Ein alternativer Index-Cluster, im Folgenden AIX, ist ein eigener VSAM-Cluster mit einer Indexkomponente und einer Datenkomponente. Die Indexkomponente speichert die Sekundärschlüssel und die Datenkomponente speichert nicht direkt die ursprünglichen Daten sondern Records, die Sekundärschlüssel und zugeordnete Primärschlüssel enthalten.

Da identische Sekundärschlüssel in mehr als einem logischen Record auftreten können, kann jedem Sekundärschlüssel in der AIX Datenkomponente eine Gruppe von Primärschlüsseln zugeordnet sein.

Die Erstellung eines AIX wird ebenfalls über das IDCAMS-Utility vorgenommen mittels des BLDINDEX-Befehls. Ein AIX kann nur nach Definition des dazugehörigen Basis-Clusters definiert werden. Der BLDINDEX-Befehl bewirkt einen sequentiellen Scan des angegebenen Basis Cluster, währenddessen Sekundärschlüsselwerte und Primärschlüssel extrahiert werden. Daraus werden die AIX-Records in der Datenkomponente des AIX-Clusters erzeugt.

Das folgende JCL-Skript legt zuerst den alternativen Index an. Der alternative Schlüssel hat eine Länge von 8 Zeichen und beginnt bei Offset 7 (direkt nach der Matrikelnummer). Damit der alternative Index funktioniert, wird daraufhin ein PATH angelegt und zum Schluss der alternative Index erstmalig erstellt. Das Wiederholen dieses Schrittes ist durch die Angabe von `UPGRADE` beim Definieren des alternativen Indexes nicht nötig. Das JCL-Skript `DEFAIX` führt die notwendigen Anpassungen durch.

```
//PRAKxxxD JOB (),NOTIFY=&SYSUID
//*
//* DEFINE VSAM ALTERNATE INDEX
//*
//DEFCLS   EXEC PGM=IDCAMS,REGION=4096K
//SYSPRINT DD SYSOUT=A
//SYSIN    DD *

  DEFINE ALTERNATEINDEX ( -
           NAME(PRAKxxx.VSAM.STUDENT.ALTINDEX) -
           RELATE(PRAKxxx.VSAM.STUDENT) -
           VOL(Z8RES1) -
           RECORDSIZE(55 55) -
           KILOBYTES(100 100) -
           KEYS(8 7) -
           UPGRADE )
  DEFINE PATH ( -
           NAME(PRAKxxx.VSAM.STUDENT.PATH) -
           PATHENTRY(PRAKxxx.VSAM.STUDENT.ALTINDEX) )
  BLDINDEX INDATASET(PRAKxxx.VSAM.STUDENT) -
           OUTDATASET(PRAKxxx.VSAM.STUDENT.ALTINDEX) -
           SORTCALL
/*
```

Um nun über den alternativen Index auf die Daten zugreifen zu können, muss das COBOL-Programm angepasst werden. Im folgenden `Unified Diff` sind die Änderungen aufgeführt, um das Quelltext des neuen COBOL-Programms `STUD2` zu erhalten:

```
@@ -17,0 +18 @@
+           ALTERNATE RECORD KEY IS STUDSERV
@@ -19 +20 @@
-           SELECT MATNR-IN
+           SELECT STUDSERV-IN
@@ -22 +23 @@
-           FILE STATUS IS MATNR-IN-CODE.
+           FILE STATUS IS STUDSERV-IN-CODE.
@@ -33 +34 @@
```

```
-␣␣␣␣␣␣␣␣FD␣MATNR-IN
+␣␣␣␣␣␣␣␣FD␣STUDSERV-IN
@@␣-38␣+39␣@@
-␣␣␣␣␣␣␣␣␣␣01␣MATNR-RECORD␣␣PIC␣X(80).
+␣␣␣␣␣␣␣␣␣␣01␣STUDSERV-RECORD␣␣PIC␣X(80).
@@␣-44␣+45␣@@
-␣␣␣␣␣␣␣␣␣␣05␣MATNR-IN-CODE␣␣PIC␣X(2).
+␣␣␣␣␣␣␣␣␣␣05␣STUDSERV-IN-CODE␣␣PIC␣X(2).
@@␣-54,3␣+55,3␣@@
-␣␣␣␣␣␣␣␣01␣WS-MATNR-IN-RECORD.
-␣␣␣␣␣␣␣␣␣␣05␣WS-MATNR-IN␣␣PIC␣X(7).
-␣␣␣␣␣␣␣␣␣␣05␣FILLER␣PIC␣X(73).
+␣␣␣␣␣␣␣␣01␣WS-STUDSERV-IN-RECORD.
+␣␣␣␣␣␣␣␣␣␣05␣WS-STUDSERV-IN␣␣PIC␣X(8).
+␣␣␣␣␣␣␣␣␣␣05␣FILLER␣PIC␣X(72).
@@␣-59,3␣+60,3␣@@
-␣␣␣␣␣␣␣␣␣␣␣␣OPEN␣INPUT␣MATNR-IN.
-␣␣␣␣␣␣␣␣␣␣␣␣READ␣MATNR-IN␣INTO␣WS-MATNR-IN-RECORD.
-␣␣␣␣␣␣␣␣␣␣␣␣DISPLAY␣"SUCHE␣"␣WS-MATNR-IN.
+␣␣␣␣␣␣␣␣␣␣␣␣OPEN␣INPUT␣STUDSERV-IN.
+␣␣␣␣␣␣␣␣␣␣␣␣READ␣STUDSERV-IN␣INTO␣WS-STUDSERV-IN-RECORD.
+␣␣␣␣␣␣␣␣␣␣␣␣DISPLAY␣"SUCHE␣"␣WS-STUDSERV-IN.
@@␣-68,2␣+69,2␣@@
-␣␣␣␣␣␣␣␣␣␣␣␣MOVE␣WS-MATNR-IN␣TO␣MATNR.
-␣␣␣␣␣␣␣␣␣␣␣␣READ␣STUD-FILE␣RECORD␣KEY␣IS␣MATNR.
+␣␣␣␣␣␣␣␣␣␣␣␣MOVE␣WS-STUDSERV-IN␣TO␣STUDSERV.
+␣␣␣␣␣␣␣␣␣␣␣␣READ␣STUD-FILE␣RECORD␣KEY␣IS␣STUDSERV.
@@␣-80␣+81␣@@
-␣␣␣␣␣␣␣␣␣␣␣␣CLOSE␣MATNR-IN.
+␣␣␣␣␣␣␣␣␣␣␣␣CLOSE␣STUDSERV-IN.
```

Alternativ können Sie natürlich das Programm `STUD2` kompletten neu erstellen und erhalten folgenden Quelltext:

```
␣␣␣␣␣␣␣IDENTIFICATION␣DIVISION.
␣␣␣␣␣␣␣*
␣␣␣␣␣␣␣*␣Programm␣zum␣Zugriff␣auf␣VSAM-Dataset
␣␣␣␣␣␣␣*
␣␣␣␣␣␣␣PROGRAM-ID.␣␣STUD.
␣␣␣␣␣␣␣/
␣␣␣␣␣␣␣ENVIRONMENT␣DIVISION.
␣␣␣␣␣␣␣*---------------------
␣␣␣␣␣␣␣CONFIGURATION␣SECTION.
␣␣␣␣␣␣␣SPECIAL-NAMES.
```

```
␣␣␣␣␣␣␣INPUT-OUTPUT␣SECTION.
␣␣␣␣␣␣␣FILE-CONTROL.
␣␣␣␣␣␣␣␣␣␣␣SELECT␣STUD-FILE
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣ASSIGN␣TO␣STUDDS
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣ORGANIZATION␣IS␣INDEXED
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣ACCESS␣IS␣DYNAMIC
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣RECORD␣KEY␣IS␣MATNR
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣ALTERNATE␣RECORD␣KEY␣IS␣STUDSERV
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣FILE␣STATUS␣IS␣FSTAT-CODE␣VSAM-CODE.
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣SELECT␣STUDSERV-IN
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣ASSIGN␣TO␣SYSIN
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣ORGANIZATION␣IS␣SEQUENTIAL
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣FILE␣STATUS␣IS␣STUDSERV-IN-CODE.
␣␣␣␣␣␣␣DATA␣DIVISION.
␣␣␣␣␣␣␣*-------------
␣␣␣␣␣␣␣FILE␣SECTION.
␣␣␣␣␣␣␣FD␣STUD-FILE
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣RECORD␣CONTAINS␣55␣CHARACTERS.
␣␣␣␣␣␣␣␣␣␣␣01␣STUDENT-RECORD.
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣05␣MATNR␣␣␣␣␣␣␣␣PIC␣X(7).
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣05␣STUDSERV␣␣␣␣␣PIC␣X(8).
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣05␣NNAME␣␣␣␣␣␣␣␣PIC␣X(20).
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣05␣VNAME␣␣␣␣␣␣␣␣PIC␣X(20).
␣␣␣␣␣␣␣FD␣STUDSERV-IN
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣RECORDING␣MODE␣F
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣BLOCK␣0␣RECORDS
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣RECORD␣80␣CHARACTERS
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣LABEL␣RECORD␣STANDARD.
␣␣␣␣␣␣␣␣␣␣␣01␣STUDSERV-RECORD␣␣␣PIC␣X(80).
␣␣␣␣␣␣␣/
␣␣␣␣␣␣␣WORKING-STORAGE␣SECTION.
␣␣␣␣␣␣␣␣01␣STATUS-AREA.
␣␣␣␣␣␣␣␣␣␣␣05␣FSTAT-CODE␣PIC␣X(2).
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣88␣I-O-OKAY␣␣␣VALUE␣ZEROES.
␣␣␣␣␣␣␣␣␣␣␣05␣STUDSERV-IN-CODE␣␣␣PIC␣X(2).
␣␣␣␣␣␣␣␣␣␣␣05␣VSAM-CODE.
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣10␣VSAM-R15-RETURN-CODE␣␣␣␣␣PIC␣9(2)␣COMP.
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣10␣VSAM-FUNCTION-CODE␣␣␣␣␣␣␣PIC␣9(1)␣COMP.
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣10␣VSAM-FEEDBACK-CODE␣␣␣␣␣␣␣PIC␣9(3)␣COMP.
␣␣␣␣␣␣␣␣01␣WS-STUDENT.
␣␣␣␣␣␣␣␣␣␣␣05␣WS-MATNR␣␣␣␣␣␣PIC␣X(7).
␣␣␣␣␣␣␣␣␣␣␣05␣WS-STUDSERV␣␣␣PIC␣X(8).
␣␣␣␣␣␣␣␣␣␣␣05␣WS-NNAME␣␣␣␣␣␣PIC␣X(20).
␣␣␣␣␣␣␣␣␣␣␣05␣WS-VNAME␣␣␣␣␣␣PIC␣X(20).
```

```
       01 WS-STUDSERV-IN-RECORD.
          05 WS-STUDSERV-IN  PIC X(8).
          05 FILLER PIC X(72).
      /
       PROCEDURE DIVISION.
           OPEN INPUT STUDSERV-IN.
           READ STUDSERV-IN INTO WS-STUDSERV-IN-RECORD.
           DISPLAY "SUCHE " WS-STUDSERV-IN.
           OPEN INPUT STUD-FILE.
           IF FSTAT-CODE NOT = "00"
               DISPLAY "OPEN INPUT VSAM FILE FS-CODE: " FSTAT-CODE
               PERFORM VSAM-CODE-DISPLAY
               STOP RUN
           END-IF.
           MOVE WS-STUDSERV-IN TO STUDSERV.
           READ STUD-FILE RECORD KEY IS STUDSERV.
           IF FSTAT-CODE NOT = "00" AND FSTAT-CODE NOT = "02"
               DISPLAY "READ STUD-FILE FS-CODE: " FSTAT-CODE
               PERFORM VSAM-CODE-DISPLAY
               STOP RUN
           END-IF.
           MOVE MATNR TO WS-MATNR.
           MOVE STUDSERV TO WS-STUDSERV.
           MOVE NNAME TO WS-NNAME.
           MOVE VNAME TO WS-VNAME.
           CLOSE STUD-FILE.
           CLOSE STUDSERV-IN.
           DISPLAY "MATNR:    " WS-MATNR.
           DISPLAY "STUDSERV: " WS-STUDSERV.
           DISPLAY "NACHNAME: " WS-NNAME.
           DISPLAY "VORNAME:  " WS-VNAME.
           STOP RUN.

       VSAM-CODE-DISPLAY.
           DISPLAY "VSAM CODE -->"
                   " RETURN: " VSAM-R15-RETURN-CODE,
                   " COMPONENT: " VSAM-FUNCTION-CODE,
                   " REASON: " VSAM-FEEDBACK-CODE.
```

Nachdem dieses ebenfalls kompiliert wurde, lässt es sich wieder über ein JCL-Skript `RUN2` starten. Dabei ist zu beachten, dass über ein zusätzliches `STUDDS1 DD`-Statement der alternative Index über den erstellten `PATH` bekannt gemacht werden muss.

```
//PRAKxxxR JOB (),NOTIFY=&SYSUID
//*
//* AUSFUEHREN DES COBOL-PROGRAMMS ZUM VSAM-ZUGRIFF
//*
//RUN      EXEC PGM=STUD2
//STEPLIB  DD DSN=&SYSUID..VSAM.LOAD,DISP=SHR
//STUDDS   DD DSN=&SYSUID..VSAM.STUDENT,DISP=SHR
//STUDDS1  DD DSN=&SYSUID..VSAM.STUDENT.PATH,DISP=SHR
//SYSIN    DD *
mam16ddd
/*
```

Das Ergebnis befindet sich nach dem `SUBMIT` des Skriptes im Joblog wieder. Eine mögliche Ausgabe können sie in Abbildung 4 sehen.

```
 SDSF OUTPUT DISPLAY PRAKxxxR JOB*****  DSID   102 LINE 1       COLUMNS 01- 80
 COMMAND INPUT ===>                                          SCROLL ===> PAGE
 SUCHE mam16ddd
 MATNR:    3333333
 STUDSERV: mam16ddd
 NACHNAME: Parziale
 VORNAME:  Lydia       00000400
******************************** BOTTOM OF DATA ********************************
```

Abbildung 4: Ergebnis des angepassten COBOL-Programms im Joblog

2 Aufgaben

Genereller Hinweis

Ersetzen Sie den Platzhalter `xxx` mit ihrer Pratikumsnummer! Dies gilt sowohl für Dataset-Namen als auch für die Inhalte der Member (JCL-Skripte, COBOL-Quelltexte, etc.). Sollten Sie sich nicht daran halten, droht Ihnen Punktabzug!

Ebenfalls sind Sie dazu angehalten nach der Abgabe ihrer Lösungen, die angelegten Joblogs zu löschen.

Aufgabe 1

Legen Sie ein neues PDS "`PRAKxxx.VSAM.CNTL`" an. Beachten Sie, dass dieser eine Größe von zwei Tracks nicht überschreitet. Empfohlene Werte:

Space units	KB	Record format	FB
Primary quantity	16	Record length	80
Secondary quantity	1	Block size	320
Directory blocks	2	Data set name	PDS

Erstellen Sie einen Member `DEFCLUST`, mit dessen Hilfe der VSAM-Cluster erstellt werden kann und führen Sie es aus. Prüfen Sie, ob der VSAM-Cluster tatsächlich angelegt wurde und belegen Sie es mit einem Screenshot (`1_DSLIST`) von `DSLIST`.

Ergebnisse:

- Screenshot 1_DSLIST.jpg (oder png)

(maximal 1 Punkt)

Aufgabe 2

Legen Sie ein neues sequenzielles Dataset "`PRAKxxx.VSAM.SEQDATA`" an. Da dieses nur sehr wenige Beispiel-Daten aufnehmen soll, reicht eine Dataset-Größe von einem Track aus. Empfohlene Werte:

Space units	TRKS	Record format	FB
Primary quantity	1	Record length	55
Secondary quantity	1	Block size	220
Directory blocks	0	Data set name	

Füllen Sie unter Nutzung des ISPF-Editors das Dataset mit einigen Beispiel-Datensätzen und stellen Sie sicher, dass sich darunter auch Ihr eigener befindet. Kopieren Sie die Daten in Ihren VSAM-Cluster.

Ergebnisse:

- Machen Sie einen Screenshot (`2_SEQDATA`), der Ihre Beispieldatensätze (inkl. Ihres Datensatzes) zeigt.
- Erstellen Sie einen weiteren Screenshot aus dem SDFS, (`2_SDSF`)der zeigt, dass die Datensätze erfolgreich in den VSAM-Cluster kopiert wurden.

(maximal 3 Punkte)

Aufgabe 3

Erstellen Sie ein weiteres PDS "`PRAKxxx.VSAM.COBOL`", das zukünftig die Quelltexte der beiden COBOL-Programme aufnehmen soll analog zu `PRAKxxx.VSAM.CNTL`.

Legen Sie in diesem einen Member "`STUD`" an, der das erste COBOL-Programm aufnimmt.

Erstellen Sie ein weiteres PDS "`PRAKxxx.VSAM.LOAD`", das zukünftig alle ausführbaren Programme aufnehmen soll (`Record format` U).

Kompilieren Sie das Programm und führen Sie es anschließend mit ihrer Matrikelnummer aus. Machen Sie einen Screenshot (`3_SDSF`), der das Suchergebnis mit Ihren Daten zeigt.

Ergebnisse:

- Screenshot 3_SUBMIT.jpg (oder png), der zeigt, dass das COBOL-Programm erfolgreich ausgeführt wurde.
- Screenshot 3_SDSF.jpg (oder png)

(maximal 5 Punkte)

Aufgabe 4

Erstellen Sie den alternativen Index auf die Studserv-Kürzel und passen Sie Ihr COBOL-Programm entsprechend an. Fertigen Sie einen Screenshot (`4_SDSF`), der das Suchergebnis mit Ihren Daten zeigt mit Hilfe des angepassten Programms.

Ergebnisse:

- Screenshot 4_SUBMIT.jpg (oder png), der zeigt, dass das VSAM-Cluster erfolgreich angepasst wurde.
- Screenshot 5_SUBMIT.jpg (oder png), der zeigt, dass das COBOL-Programm erfolgreich ausgeführt wurde.
- Screenshot 4_SDSF.jpg (oder png)

(maximal 3 Punkte)

Abgabe:
Erstellen Sie eine `pdf`-Datei in das Sie **alle** Ergebnisse zu den Aufgaben einfügen und laden Sie diese bei Moodle in den Abgabeordner hoch.

Dateiname: `ECG3_<Namen beider Teammitglieder>`

Für das Erreichen der maximalen Punktzahl müssen bei der Praktikumsabgabe neben der vollständigen Bearbeitung der Aufgaben, auch Kenntnisse und Verständnis der zugrunde liegenden Themen dieses Arbeitsblattes bestätigt werden.

3 Quellen

Bei diesem Dokument handelt es sich um eine grundlegende Überarbeitung des alten Tutorials (Tut17) zu der Vorlesung “Einführung in z/OS” der Universität Leipzig und Tübingen.

Die Aufgaben-Verschönerung wurde von `http://www.texample.net/tikz/examples/framed-tikz/` übernommen.