Erstellen und Benutzen von VSAM-Datasets

Dies ist ein einführendes Tutorial zur Benutzung von VSAM-Datasets.
Dieses Tutorial behandelt „Key Sequential Datasets“ (KSDS).

Das Tutorial findet komplett unter TSO statt. Es werden Erfahrungen im Umgang mit ISPF (Umgang mit Datasets), dem ISPF-Editor und SDSF vorausgesetzt.

Als Beispiel dient folgendes:

Es sollen für die Universität Matrikelnummern, Studserv-Kennung sowie Vor- und Nachname in einem VSAM Dataset gespeichert werden. Dabei soll in einem Programm einmal über die Matrikel-Nummer und einmal über das Studserv-Kennung ein Zugriff auf die restlichen Daten erfolgen.

Für diesen Zweck soll zuerst ein VSAM-Dataset angelegt werden, der die Daten beinhaltet und einen Index auf die Matrikelnummer anlegt. Dieser wird danach mit den Daten der Studenten gefüllt werden.

Um auf die Daten zuzugreifen, ist ein COBOL-Programm zu schreiben. Das Programm soll über einen JCL-Script gestartet werden und dann als Batch-Job laufen.

Anschließend wird der VSAM-Dataset um einen alternativen Index – die Studserv-Kennung – erweitert und ein weiteres COBOL-Programm geschrieben (bzw. das vorhandene angepasst), das über diesen Index die Daten sucht.

Das Tutorial besteht aus den folgenden drei Teilen:

1. **Erstellen des VSAM-Datasets**
2. **Schreiben des COBOL-Programms**
3. **Erweitern um alternativen Index**

Inhalt

Für die Durchführung des Tutorials erstellen wir eine ganze Reihe von Datasets und deren Members.Dies sind spezifisch:

PRAKT20.VSAM.STUDENT	**VSAM Cluster**
PRAKT20.VSAM.STUDENT.DATA	**Data Component des VSAM Cluster**
PRAKT20.VSAM.STUDENT.INDEX	**Index Component des VSAM Cluster**
PRAKT20.VSAM.STUDENT.ALTINDEX	**VSAM Cluster**
PRAKT20.VSAM.STUDENT. ALTINDEX.DATA	**Data Component des VSAM Cluster**
PRAKT20.VSAM.STUDENT.ALTINDEX.INDEX	**Index Component des VSAM Cluster**
PRAKT20.VSAM.STUDENT.PATH	**Pfad zum VSAM Cluster**
PRAKT20. VSAM.SEQDATA	**Sequentieller Dataset für Daten Eingabe**
PRAKT20. VSAM.COBOL(STUD)	**Cobol Programm # 1**
PRAKT20. VSAM.COBOL(STUD2)	**Cobol Programm # 2**
PRAKT20.VSAM.CNTL(DEFCLUST)	**JCL Script**
PRAKT20. VSAM.CNTL(REPRO)	**JCL Script**
PRAKT20. VSAM.CNTL(COMPILE)	**JCL Script**
PRAKT20. VSAM.CNTL(RUN)	**JCL Script**
PRAKT20. VSAM.CNTL(DEFAIX)	**JCL Script**
PRAKT20. VSAM.CNTL(COMP2)	**JCL Script**
PRAKT20. VSAM.CNTL(RUN2)	**JCL Script**
PRAKT20.VSAM.LOAD(STUD)	**Load Module**
PRAKT20.VSAM.LOAD(STUD2)	**Load Module**

1. Einführung

1.1 VSAM

z/OS benutzt den Begriff Access Method um einen bestimmten Dataset Typ zu bezeichnen. Unter Linux wäre der Begriff „File System“ äquivalent. Genauso wie ein Linux System mit mehreren File Systemen arbeiten kann (extfs3, extfs4, ReiserFS), existieren auch unterschiedliche Access Methods unter z/OS. VSAM (Virtuell Storage Access Method) ist die wichtigste z/OS Access Method.

Der Begriff Virtual Storage Access Method (VSAM) bezeichnet sowohl den Dataset Typ (Organisation) als auch die Verfahren (access method) um unterschiedliche Benutzer Daten Typen zu managen und 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.

VSAM wird in erster Linie für Anwendungsdaten eingesetzt. Es wird nicht für die Speicherung von Quellcode, JCL, oder ausführbaren Programme benutzt. VSAM Datasets können nicht ohne weiteres mit ISPF editiert oder wiedergegeben werden. Spezielle VSAM Editoren sind jedoch von mehreren unabhängigen Herstellern erhältlich.

VSAM Records können auf 4 unterschiedliche Arten organisiert und benutzt werden: Entry-Sequenced, Key-Sequenced, Relative Record (direkt), oder linear. Hiervon ist der Key Sequence Dataset (KSDS) am gebräuchlichsten.

Die (logischen) Records in einer VSAM Datei können eine unterschiedliche Länge haben.

Ein neuer VSAM Dataset wird mit Hilfe einer z/OS System Utility „IDCAMS“ erstellt.

Literatur: IBM Redbook: “VSAM Demystified”. September 2003, SG24-6105-01, http://www.redbooks.ibm.com/redbooks/pdfs/sg246105.pdf

11.2 Blocking

In der Anfangszeit der Mainframe Entwicklung benutzte man „Basic“ Access Methods. Wenn im Anwendungsprogramm ein READ oder WRITE Befehl ausgeführt wurde, wurde durch das Betriebssystem ein einziger Record von der Festplatte gelesen oder auf die Festplatte geschrieben.

Sehr bald ging man zu „Queued“ Access Methods über. Bei der Ausführung eines READ Befehls wurde von der Festplatte immer eine Gruppe benachbarter Records gelesen und in einen entsprechend größeren Buffer Bereich des Hauptspeichers gelesen. Mit etwas Glück befand sich der gewünschte Record bei einem folgenden READ Befehl bereits im Hauptspeicher, und man konnte sich den aufwendigen Lesevorgang vom Plattenspeicher ersparen.

Hierzu fasste man mehrere Records (auch als „logische“ Records bezeichnet) zu einem Block (auch als „physische Records bezeichnet) zusammen. Beim Zugriff auf die Festplatte wurde immer ein Block an Stelle eines einzelnen Records gelesen.

In unserem Tutorial 1a haben wir erstmalig einen Data set allocated. Dort machten wir diese Angaben:

```
Primary quantity  . . 16              (In above units)
Secondary quantity    1               (In above units)
Directory blocks  . . 2               (Zero for sequential data set) *
Record format . . . . FB
Record length . . . . 80
Block size  . . . . . 320
Data set name type    PDS             (LIBRARY, HFS, PDS, LARGE, BASIC, *
```

Wir haben die (logische) Record Länge mit 80 Byte definiert, und haben eine Block (physischer Record) Größe von 320 Bytes festgelegt. Jeder Block nimmt also 4 logische Records auf, und bei einem Lese Zugriff auf den Festplattenspeicher werden immer 320 Bytes ausgelesen.

Für eine Diskussion über optimale Block Größtem siehe Band 1 Abschnitt 5.3.4.

1.3 VSAM Dataset Terms

VSAM Datasets werden andersartig als non-VSAM Datasets auf dem Plattenspeicher abgespeichert..

VSAM speichert Records in Blöcken, die als Control Intervals bezeichnet werden Ein Control Interval (CI) ist ein zusammenhängender (contiguous) Bereich auf einem DASD (disk drive), welcher sowohl Records als auch Steuerinformation speichert. Daten werden zwischen Hauptspeicher und Plattenspeicher als ganze Control Intervals bewegt. Die Größe der CIs kann von einem VSAM Dataset zum nächsten VSAM Dataset unterschiedlich sein; alle CIs innerhalb eines spezifischen VSAM Datasets haben jedoch die gleiche Länge. Eine sehr häufig benutzte Control Interval Größe ist 4 KByte, identisch mit der Größe eines 4 KByte Virtual Storage Page Frames.

Ein Control Interval besteht aus

- **mehreren logischen Records, plus einem**
- **Control Interval Definition Field (CIDF) sowie mehreren**
- **Record Definition Fields (RDFs).**

In so fern unterscheidet sich ein CI von den Blöcken (physischen Records) anderer Dataset Access Methods, bei denen ein Block lediglich eine Aneinanderreihung von (logischen) Records enthält.

Mehrere Control Intervals in einem VSAM Dataset werden in einem zusammenhängender (contiguous) Bereich auf einem DASD (disk drive) zusammengefasst, der als Control Area bezeichnet wird. Eine Control Area hat häufig die Größe eines Zylinders (15 Spuren) einer 3390 Festplatte, oder 849960 Bytes.

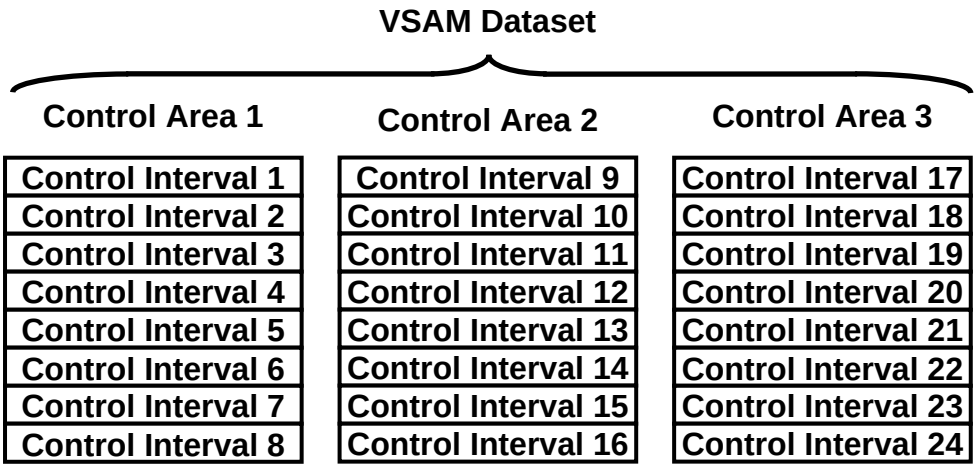


Abb.1.1 : Beispiel eines VSAM Datasets, der aus 24 Control Intervallen und 3 Control Areas besteht.

Ein VSAM Dataset kann aus vielen Control Areas bestehen. Mit einer Control Interval Größe von 4 KByte ist die maximale Größe eines VSAM Datasets von 16 TByte möglich.

1.4 Key Sequenced Dataset (KSDS)

Jeder VSAM Record besteht aus mehreren Feldern. Eines dieser Felder kann als Schlüsselfeld (Key Field) definiert werden.

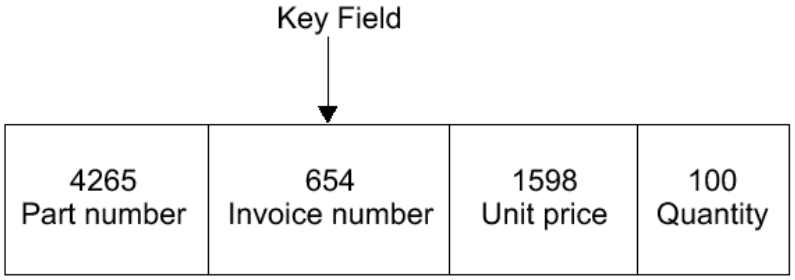


Abb. 1.2

In diesem Beispiel besteht jeder Record eines VSAM Datasets aus 4 Feldern. Die Invoice Number wird als Key Field benutzt. Für den hier gezeigten Record hat das Key Field den Wert 654.

Das Key Field muss sich in der gleichen Position in jedem Datensatz eines Key Sequenced Dataset befinden (das zweite Feld in der Abbildung oben). Jeder Datensatz Key muss eindeutig sein. Der Wert des Keys kann nachträglich nicht geändert werden; jedoch kann der gesamte Datensatz gelöscht werden.

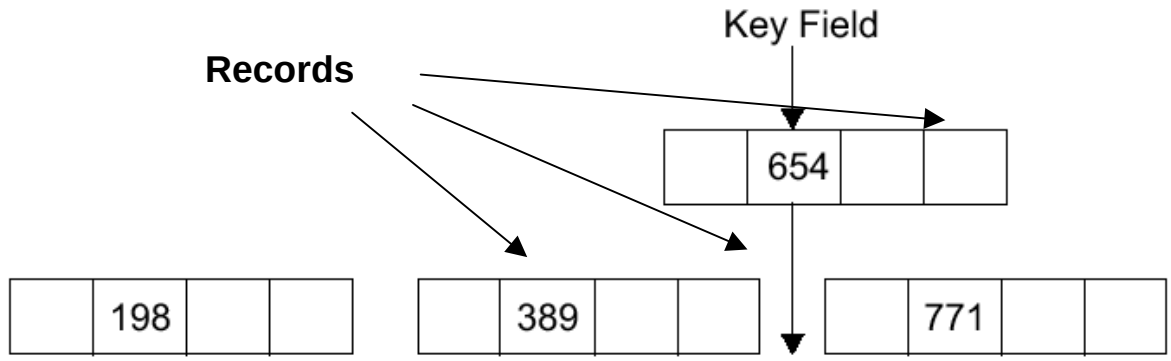
In einem Key-Sequenced Dataset sind logische Records in dem Dataset in aufsteigender Reihenfolge entsprechend der Werte in dem Key Field angeordnet. Der Key enthält einen eindeutigen Wert, z.B. eine Mitarbeiter Nummer oder eine Rechnungs Nummer. Dieser Wert bestimmt die Sortier- (collating) Position des Records in dem Dataset.

Werden neue Records erzeugt, werden diese an der richtigen Sortierstelle eingeschoben. Vorhanden Records werden verschoben um die korrekte Reihenfolge zu erhalten.

In einem KSDS können Records entsprechend ihrer Key Werte sowohl sequentiell als auch zufallsbedingt verarbeitet werden. Die Vorteile der KSDS sind:

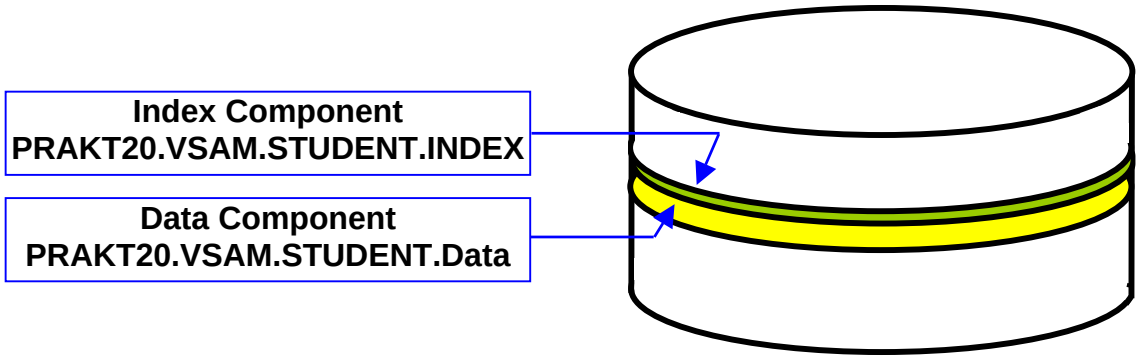
- **Eine sequentielle Verarbeitung ist für das Abrufen von Datensätzen in der sortierten Form nützlich.**
- **Eine zufällige oder direkte Verarbeitung der Rekords ist nützlich bei interaktiven Online-Anwendungen.**

Datensätze in einer KSDS werden in Key Sequence gespeichert. Das Key Field der Records bestimmt die Reihenfolge, in der die Datensätze gespeichert werden.



Wenn ein neuer Record dem Dataset hinzugefügt wird, wird es in der Sortierfolge seines Keys eingefügt. In dem gezeigten Beispiel enthält das VSAM-Dataset Control Interval drei Records mit den Key-Werten 198, 389 und 771. Ein neuer Record mit dem Key-Wert 654 wird innerhalb des CIs nach dem Record mit dem Key 389 und vor dem Record mit dem Key 771 eingefügt.

1.5 Index – und Data Component



Auf einem Plattenspeicher nimmt ein Key Sequenced Dataset (KSDS) zwei lineare Speicherbereiche ein, sogenannt Komponenten. Es gibt zwei Arten von Komponenten, die Daten-Komponente und die Index-Komponente. Die Daten Komponente ist der Teil einer VSAM-Datet, welcher die Daten Records enthält. Alle VSAM Datasets haben eine Daten-Komponente. Andere VSAM Organisationen, zum Beispiel Entry Sequenced Datasets (ESDS), haben nur die Daten-Komponente.

- **Die Datensätze (Records) der Datenkomponente werden in einem linearen Bereich auf der Plattenspeicher gespeichert, und die Index-Datensätze werden in einer zweiten linearen Raum auf dem Datenträger gespeichert.**

- **Die Index-Komponente ist eine Sammlung von Records (Index logical Records), welche**
  - **Data Keys der Records der Datenkomponente enthalten, sowie deren**
  - **Adressen (RBA, Relative Byte Address).**

- **Die Data Keys werden von dem Key-Feld in jedem Datenrecord kopiert. Mit Hilfe des Index ist VSAM der Lage, einen logischen Rekord aus dem Daten-Komponente abzurufen, wenn eine Anfrage für einen Record mit einem bestimmten Key gemacht wird.**

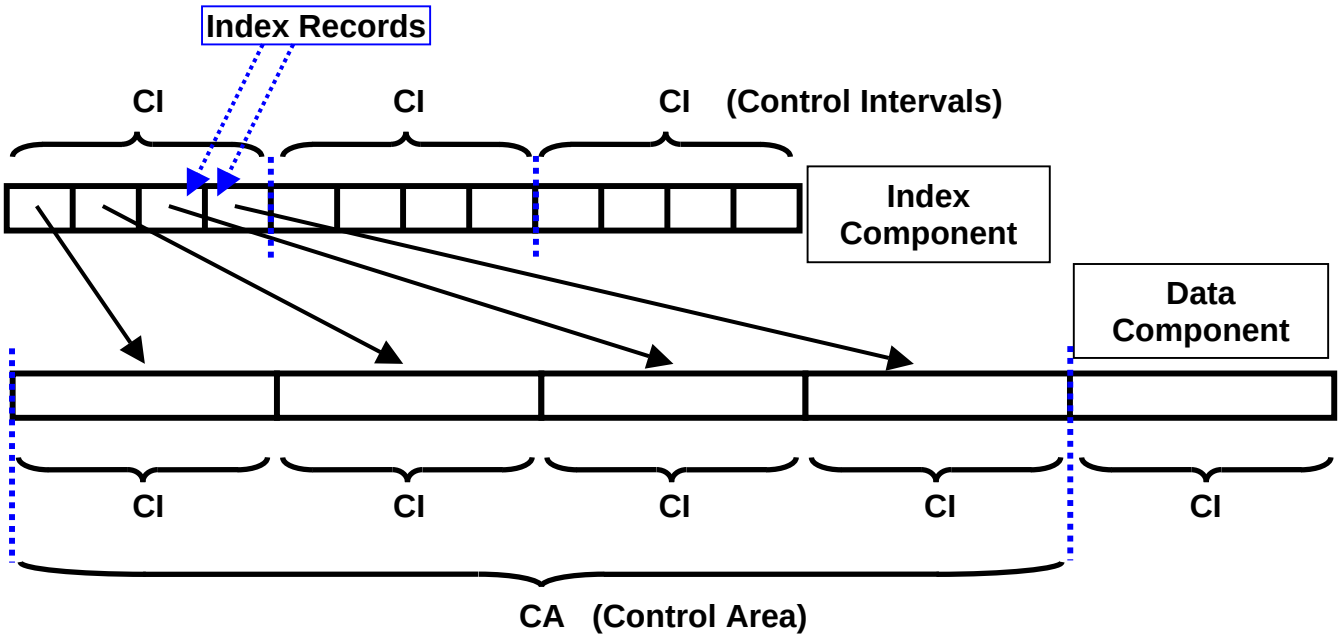
Die Datenkomponente und die Index-Komponente sind wirklich zwei unabhängige VSAM Datasets.

Ein VSAM-Cluster ist die Kombination der Daten Komponente (Dataset) und der Index-Komponente (Dataset) eines KSDS. Das Cluster Konzept vereinfacht die VSAM Verarbeitung. Es bieten eine Möglichkeit, Index und Daten-Komponenten als eine Einheit zu behandeln, und mit einem eigenen Namen zu katalogisieren. Es kann auch jede Komponente einen eigenen Namen haben. Dies ermöglicht es, die Daten Komponente getrennt von der Index-Komponente zu verarbeiten.

Jetzt kommt der entscheidende VSAM Frage: "Was in VSAM entspricht einem z/OS Dataset?" Die beste Antwort ist, es hängt von den Umständen ab. Wenn Sie zum Beispiel den Cluster-Namen in einem DD-Anweisung eines JCL-Script benutzen, dann entspricht der Cluster dem Dataset. Wenn Sie sich auf eine Komponente beziehen, dann entspricht diese dem Dataset.

(Die Bedeutung des Begriffs "Cluster", im Kontext mit VSAM, ist nicht identisch mit der Bedeutung des Begriffs "Cluster" in einer parallel Processing Configuration.)

In unserer VSAM Übungsaufgabe definieren wir einen VSAM Cluster mit dem Namen PRAKT20.VSAM.STUDENT, der aus einer Data Component PRAKT20.VSAM.STUDENT.DATA sowie einer Index Component PRAKT20.VSAM.STUDENT.INDEX besteht.



Die Index-Komponente besteht aus mehreren CIs (und wahrscheinlich auch aus mehreren CAs). Jeder CI in der Index-Komponente besteht aus mehreren Index Records, wobei jeder Index Record in der Index-Komponente auf ein CI in der Daten Komponente zeigt.

In dem oben gezeigten Beispiel enthält jedes CI in der Index-Komponente 4 Records, und jede CA in der Data Component enthält 4 CIs.

1.6 Multilevel Index Component

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

Die Suche in einem großen Index kann sehr zeitaufwendig sein. Ein mehrstufiger Index wird aufrechterhalten, um die Index-Suche zu verkürzen. VSAM teilt die Index Komponente in zwei Teile auf: Sequence-Set und Index-Set. Die unterste Ebene der Index Komponente wird als Sequence-Set bezeichnet. Die Pointer in der untersten Ebene zeigen direkt (mit Hilfe einer RBA) auf ein Kontroll-Intervall (CI) innerhalb einer Control Area (CA) der Daten Komponente. Der Sequence Set enthält

- **einen Index-Eintrag für jedes CI in der Daten Komponente, und damit auch**
- **ein Index CI für jedes CA in der Daten-Komponente.**

Bei kleinen VSAM Datasets würde die Index Komponente nur aus einem Sequence Set bestehen. Größere Index Sets unterhalten als Bestandteil ihrer Index Komponente eine oder mehrere Index Ebenen zusätzlich zu dem Sequence Set. Man bezeichnet die Ebenen oberhalb des Sequence Set als den Index Set. Er kann beliebig viele Ebenen enthalten. Sequence-Set und Index Set bilden zusammen die Index-Komponente eines KSDS.

Ein Eintrag in einem Index Set Datensatz enthält den höchstmöglichen Key in einem Index-Datensatz in der nächst tieferen Ebene, und einen Zeiger auf den Anfang dieses Index-Datensatzes. Die höchste Ebene des Index enthält immer einen einzelnen Index CI.

Selbst-Test

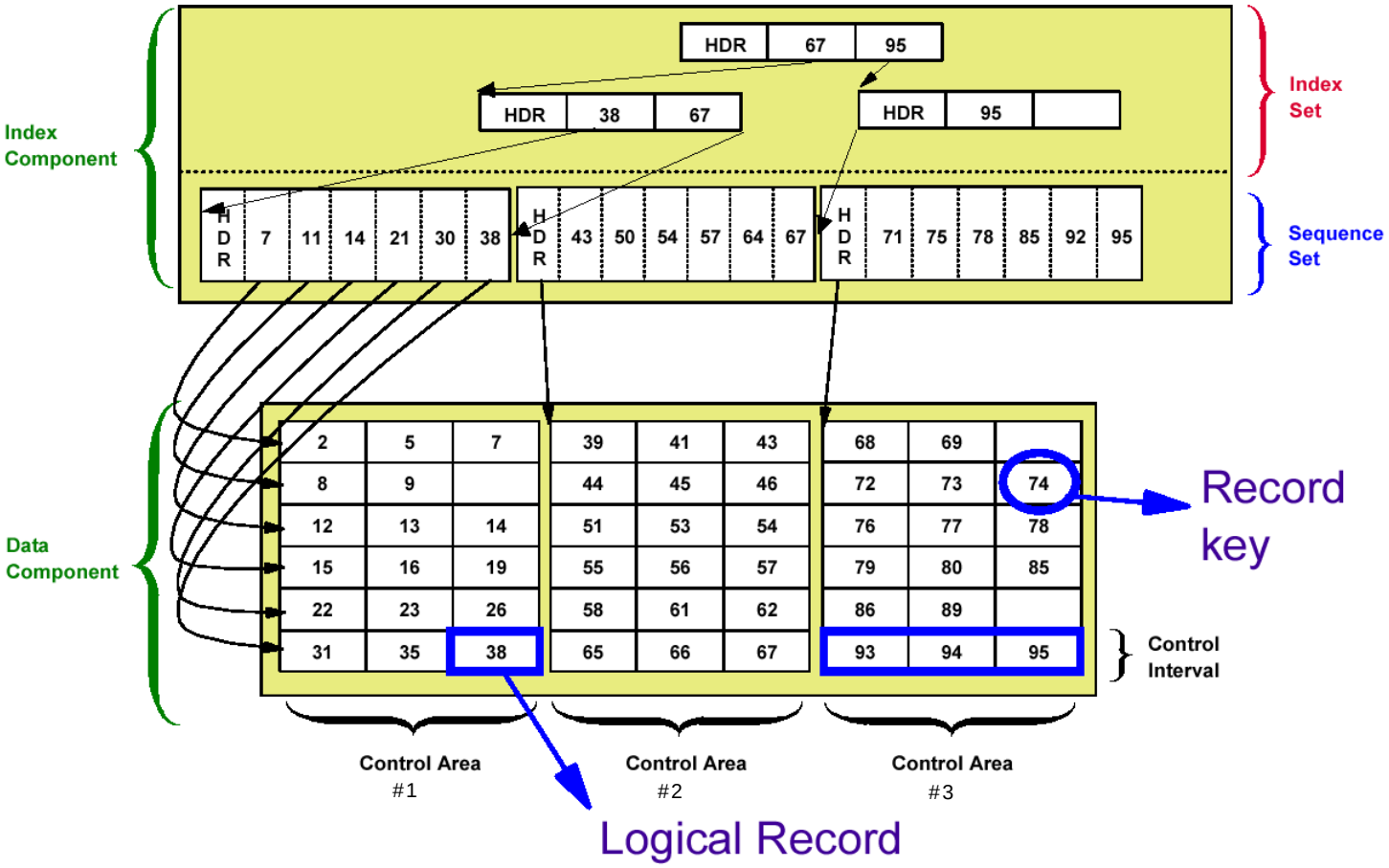
- **Sind der (die) Index Sets eigene Datasets ?**

1.7 Beispiel

Im dem unten gezeigten Beispiel, besteht die Daten Komponente eines VSAM-Key Sequenced Dataset (KSDS) aus 3 Control Areas. Jede CA besteht aus 6 Kontrollintervallen, und jedes CI speichert genau 3 Records. Es kann nützlich sein, wenn Sie die Seite mit dem Beispiel ausdrucken um den folgenden Text besser zu verstehen.

Der Sequence Set in der Index-Komponente enthält 3 CIs, eine CI für jede CA in der Daten Komponente. Jedes CI in dem Sequence Set enthält 6 Records, einen Datensatz für jedes CI in der Daten-Komponente. Jeder Sequence Set Datensatz enthält den höchsten Key in dem zugehörigen Daten-Komponente CI.

Die erste CI in der Daten-Komponente hat die Schlüssel 2, 5 und 7. Der erste Datensatz in der Sequenz-Set enthält einen Zeiger (RBA) an den Datensatz mit dem Schlüssel 7 in der Daten-Komponente.



Das zweite CI in der Daten-Komponente hat die Keys 8, 9, und einen freien Platz. Das dritte CI in der Daten-Komponente hat die Keys 12, 13 und 14. Wenn der freie Speicherplatz in dem zweiten CI in der Daten Komponente später gefüllt wird, kann ihr Key einen Wert nicht höher als 11 haben (oder es wäre an anderer Stelle platziert werden). Deshalb enthält der zweite Datensatz in dem Sequence Set einen Zeiger (RBA) auf einem potentiellen Datensatz mit dem Schlüssel 11 in der Daten-Komponente.

Für die zweite CA in der Daten-Komponente enthält der Sequence Set eine neue CI. Der erste Datensatz des CI enthält den Wert 43, weil das der Schlüssel des letzten Datensatzes in CI Nr. 1 von CA Nr. 2 in der Daten-Komponente ist.

Das letzte CI in der ersten CA der Datenkomponente hat die Keys 31, 36 und 38. Die Index Ebene unmittelbar über dem Sequenz-Set enthält einen Indexeintrag für jedes Sequence Set Controll-Intervall. Damit enthält der erste Datensatz in der ersten CI des Index Set den Wert 38. Der zweite Datensatz in dem ersten CI des Index Set enthält den Wert 67. Dies ist der höchste Key in CA 2 der Daten-Komponente.

Die unterste Ebene des Index Set enthält 2 CIs, jeweils ein CI für die beiden CAs 1 und 2 der Daten-Komponente, und ein weitere CI für CA 3 der Daten-Komponente. Diese beiden CIs werden durch eine zweite Ebene des Index Sets adressiert.

Alle Einträge werden in aufsteigender Key Reihenfolge gehalten.

Selbst-Test

- **Warum legen Sie zwei Datasets an ?**
- **Haben beide Datasets die gleiche Record Struktur ?**
- **Was bedeutet die Bezeichnung „Cluster“ ?**

2. Erstellen des VSAM-Datasets

Für das Erstellen eines VSAM-Datasets gibt es mehrere Möglichkeiten. Unter z/OS existiert ein Hilfsprogramm IDCAMS für das Anlegen von VSAM Datasets. Genauere Details der Befehle von IDCAMS kann man in „DFSMS Access Method Services for Catalogs“ (SC26-7394-03) S. 5 nachlesen.

In diesem Tutorial wird der Dataset mit einem JCL-Script unter Benutzung von IDCAMS erstellt.

Der erste Schritt bei der Generierung einer neuen Relationalen Datenbank ist die Erstellung eines Datenbank Schemas. Im Falle von VSAM muss als erster Schritt ein VSAM-Cluster angelegt werden. Weiterhin muss das (logische) Record Format definiert werden. Wir entscheiden uns für ein fixed length Record Format. Ein Record soll aus 4 Feldern bestehen:

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

Damit hat ein Record 55 Zeichen. Die Daten werden in der oben angegebenen Reihenfolge im Record gespeichert.

Zum Bearbeiten der nachfolgenden Aufgaben können Sie die Datei "vsam.zip" nutzen. Diese kann unter "Datasets zum Tutorial" von der gleichen Web-Site heruntergeladen werden wie dieses Tutorial (Vorsicht: ältere Version !!!).

Bei der Anlage aller Datasets beachten Sie bitte, dass Sie für jeden Dataset nur so viel Plattenspeicher reservieren wie Sie auch benötigen. Ein oder zwei Tracks reichen in der Regel aus.

Aufgabe: *Legen Sie einen Partitioned Dataset <Ihre User-ID>.VSAM.CNTL (im FB 80-Format) an, der zukünftig alle JCL-Scripte aufnehmen soll. Beachten Sie, dass dieser eine Größe von 2 Tracks nicht überschreitet.*

Aufgabe: *Erstellen Sie im Dataset <Ihre User-ID>.VSAM.CNTL einen Member DEFCLUST, mit dessen Hilfe der VSAM-Cluster erstellt werden kann.*

Aufgabe: *Editieren Sie in den Member DEFCLUST das JCL-Script DEFCLUST (Abb. 2.1) und führen Sie es aus (sub).*

Aufgabe: *Prüfen Sie, ob der VSAM-Cluster tatsächlich angelegt wurde*

Selbst-Test

- **Aus welchen Data Sets besteht der „Cluster“ ?**

```
//PRAKT20D JOB (),NOTIFY=&SYSUID
//*
//* DEFINE VSAM CLUSTER
//*
//DEFCLS EXEC PGM=IDCAMS,REGION=4096K
//SYSPRINT DD SYSOUT=A
//SYSIN DD *

  DELETE PRAKT20.VSAM.STUDENT

  DEFINE CLUSTER ( -
         NAME(PRAKT20.VSAM.STUDENT) -
         VOL(SMS009) -
         RECORDSIZE(55 55) -
         RECORDS(10 10) -
         KEYS(7 0) -
         INDEXED ) -
       DATA ( -
         NAME(PRAKT20.VSAM.STUDENT.DATA) ) -
       INDEX ( -
         NAME(PRAKT20.VSAM.STUDENT.INDEX) )
/*
```

Das Programm IDCAMS wird aufgerufen.

Abbildung 2.1 : JCL-Script zum Erstellen des VSAM-Clusters

Die beiden Parameter bei RECORDSIZE geben die mittlere und die maximale Größe eines Records an, die in diesem Fall gleich groß sind. Wir wählen genau 55, weil ja ein Datensatz - bestehend aus Matrikel-Nr, Studserv-Kennung, Vor- und Zuname - genau 55 Zeichen lang ist.

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.

Selbst-Test

- **Was ist IDCAMS ?**
- **Benutzen Sie IDCAMS für die Erstellung von PRAKT20.VSAM.SEQDATA ?**
- **Warum arbeiten Sie mit einer Record Länge von 55 Bytes ?**

Als nächstes soll der Cluster mit Studenten-Daten gefüllt werden. Diese später in den VSAM-Cluster zu kopierenden Daten erstellen wir zunächst in einem sequentiellen Dataset.

Aufgabe: *Legen Sie diesen sequentiellen Dataset an. Er sollte 55 Bytes/Record haben (Record-Format FB) und den Namen PRAKT20.VSAM.SEQDATA erhalten. Da dieser nur sehr wenige Beispiel-Daten aufnehmen soll, reicht eine Dataset-Größe von 1 Track.*

Aufgabe: *Füllen Sie unter Nutzung des ISPF-Editors den gerade angelegten Dataset mit einigen Beispiel-Datensätzen (siehe Abbildung 2.2). Sie können einige Fantasie-Datensätze verwenden, doch einer der Datensätze muss wahrheitsgemäß Ihre persönlichen Daten enthalten: Ihr eigener Datensatz.*

```
  File  Edit  Confirm  Menu  Utilities  Compilers  Test  Help
------------------------------------------------------------------------------
EDIT       PRAKT20.VSAM.SEQDATA                              Data set saved
****** ***************************** Top of Data ******************************
==MSG> -Warning- The UNDO command is not available until you change
==MSG>           your edit profile using the command RECOVERY ON.
000001 1187579mai91jhzMichaelsen          Nils
000002 1207747mai02wdrMueller             Georg
000003 1207864mai03werFriedrich           Boris
000004 1213435mai03fgtHeinrich            Albrecht
000005 1309745mai03hgeFischer             Katrin
****** **************************** Bottom of Data ****************************









Command ===>                                                 Scroll ===> 0001
 F1=Help      F3=Exit      F5=Rfind     F6=Rchange  F12=Cancel
```

Abbildung 2.2
PRAKT20.VSAM.SEQDATA mit 5 Datensätzen füllen

Jeder Datensatz besteht aus 4 Feldern mit einer Länge von jeweis 7, 8, 20 und 20 Bytes

Selbst-Test

- **Warum schreiben Sie diese Daten in PRAKT20.VSAM.SEQDATA ?**
- **Ist PRAKT20.VSAM.SEQDATA ein VSAM Dataset ?**
- **Falls ja, warum, falls nein, warum nicht ?**

Aufgabe: *Legen Sie im Partitioned Dataset „PRAKT20.VSAM.CNTL“ den Member REPRO an, ein JCL-Script zum Kopieren der Daten aus dem sequentiellen Dataset in den VSAM-Cluster (Abbildung 2.3).*

Aufgabe: *Kopieren Sie die Studenten-Daten hinüber, indem Sie mittels SUB das JCL-Script ausführen.*

```
//PRAKT20R JOB (),NOTIFY=&SYSUID
//*
//* COPY SEQENTIAL DATASET INTO VSAM CLUSTER
//*
//DEFCLS   EXEC PGM=IDCAMS,REGION=4096K
//SEQDD    DD DSN=PRAKT20.VSAM.SEQDATA,DISP=SHR
//VSAMDD   DD DSN=PRAKT20.VSAM.STUDENT,DISP=SHR
//SYSPRINT DD SYSOUT=A
//SYSIN    DD *
  REPRO    INFILE(SEQDD) -
           OUTFILE(VSAMDD)
/*
```

Abbildung 2.3: Inhalt des Members REPRO

Selbst-Test

- **REPRO benutzt die beiden Parameter SEQDD und VSAMDD. Was bezeichnen diese ?**

3. Schreiben des COBOL-Programms

Um auf den VSAM-Dataset zuzugreifen, erstellen wir ein COBOL-Programm.

Aufgabe: *Erstellen Sie einen weiteren Partitioned Dataset <Ihre User-ID>.VSAM.COBOL, der zukünftig den Code der beiden Cobol-Programme aufnehmen soll (Record length = 80).*

Aufgabe: *Legen Sie in diesem einen Member "STUD" mit dem im folgenden angegebenen Programm-Code an.*

Es folgt der Quelltext des Cobol-Programms:

```
000100 IDENTIFICATION DIVISION.
000200*
000300* Programm zum Zugriff auf VSAM-Dataset
000400*
000500 PROGRAM-ID.  STUD.
000600/
000700 ENVIRONMENT DIVISION.
000800*--------------------
000900 CONFIGURATION SECTION.
001000 SPECIAL-NAMES.
001100 INPUT-OUTPUT SECTION.
001200 FILE-CONTROL.
001300     SELECT STUD-FILE
001400      ASSIGN TO STUDDS
001500      ORGANIZATION IS INDEXED
001600      ACCESS IS DYNAMIC
001700      RECORD KEY IS MATNR
001800      FILE STATUS IS FSTAT-CODE VSAM-CODE.
001810     SELECT MATNR-IN
001820      ASSIGN TO SYSIN
001830      ORGANIZATION IS SEQUENTIAL
001840      FILE STATUS IS MATNR-IN-CODE.
001900 DATA DIVISION.
002000*-------------
002100 FILE SECTION.
002200 FD STUD-FILE
002300     RECORD CONTAINS 55 CHARACTERS.
002400   01 STUDENT-RECORD.
002500     05 MATNR       PIC X(7).
002600     05 STUDSERV    PIC X(8).
002700     05 NNAME       PIC X(20).
002800     05 VNAME       PIC X(20).
002810 FD MATNR-IN
002820     RECORDING MODE F
002830     BLOCK 0 RECORDS
002840     RECORD 80 CHARACTERS
002850     LABEL RECORD STANDARD.
002860   01 MATNR-RECORD  PIC X(80).
002900/
003000 WORKING-STORAGE SECTION.
003100 01 STATUS-AREA.
```

```
003200   05 FSTAT-CODE PIC X(2).
003300     88 I-O-OKAY  VALUE ZEROES.
003310   05 MATNR-IN-CODE  PIC X(2).
003400   05 VSAM-CODE.
003500     10 VSAM-R15-RETURN-CODE    PIC 9(2) COMP.
003600     10 VSAM-FUNCTION-CODE      PIC 9(1) COMP.
003700     10 VSAM-FEEDBACK-CODE      PIC 9(3) COMP.
003800 01 WS-STUDENT.
003900   05 WS-MATNR     PIC X(7).
004000   05 WS-STUDSERV  PIC X(8).
004100   05 WS-NNAME     PIC X(20).
004200   05 WS-VNAME     PIC X(20).
004210 01 WS-MATNR-IN-RECORD.
004220   05 WS-MATNR-IN  PIC X(7).
004230   05 FILLER       PIC X(73).
004300/
004400 PROCEDURE DIVISION.
004500     OPEN INPUT MATNR-IN.
004600     READ MATNR-IN INTO WS-MATNR-IN-RECORD.
004700     DISPLAY "SUCHE " WS-MATNR-IN.
004800     OPEN INPUT STUD-FILE.
004900     IF FSTAT-CODE NOT = "00"
005000       DISPLAY "OPEN INPUT VSAM FILE FS-CODE: " FSTAT-CODE
005100       PERFORM VSAM-CODE-DISPLAY
005200       STOP RUN
005300     END-IF.
005310     MOVE WS-MATNR-IN TO MATNR.
005400     READ STUD-FILE RECORD KEY IS MATNR.
005500     IF FSTAT-CODE NOT = "00" AND FSTAT-CODE NOT = "02"
005600       DISPLAY "READ STUD-FILE FS-CODE: " FSTAT-CODE
005700       PERFORM VSAM-CODE-DISPLAY
005800       STOP RUN
005900     END-IF.
006000     MOVE MATNR TO WS-MATNR.
006100     MOVE STUDSERV TO WS-STUDSERV.
006200     MOVE NNAME TO WS-NNAME.
006300     MOVE VNAME TO WS-VNAME.
006310     CLOSE STUD-FILE.
006320     CLOSE MATNR-IN.
006400     DISPLAY "MATNR:    " WS-MATNR.
006500     DISPLAY "STUDSERV: " WS-STUDSERV.
006600     DISPLAY "NACHNAME: " WS-NNAME.
006700     DISPLAY "VORNAME:  " WS-VNAME.
006800     STOP RUN.
006810
006900 VSAM-CODE-DISPLAY.
007000     DISPLAY "VSAM CODE -->"
007100             " RETURN: " VSAM-R15-RETURN-CODE,
007200             " COMPONENT: " VSAM-FUNCTION-CODE,
007300             " REASON: " VSAM-FEEDBACK-CODE.
```

Abbildubg 3.1
Cobol Quelltext

Aufgabe: *Versehen sie den obigen Cobol Code großzügig mit Kommentarzeilen und senden sie das Ergebnis an Ihren Betreuer*

Aufgabe: *Erstellen Sie einen weiteren Partitioned Dataset <Ihre User-ID>.VSAM.LOAD, der zukünftig alle ausführbaren Programme aufnehmen soll (Record Format "U").*

Aufgabe: *Um das Cobol Programm zu compilieren, existiert ein Member COMPILE (Abbildung 3.2). Legen Sie auch diesen in <Ihre User-ID>.VSAM.CNTL an, modifizieren Sie ihn und wenden Sie anschließend SUB auf ihn an.*

```
//PRAKT20C JOB (),NOTIFY=&SYSUID
//*
//* COMPILIEREN DES COBOL-PROGRAMMS ZUM VSAM-ZUGRIFF
//*
//COBCOMP EXEC IGYWCL
//COBOL.SYSIN  DD DSN=PRAKT20.VSAM.COBOL(STUD),DISP=SHR
//LKED.SYSLMOD DD DSN=PRAKT20.VSAM.LOAD,DISP=SHR
//LKED.SYSIN   DD *
 NAME STUD(R)
/*
```

Abbildung 3.2: JCL-Script zum Compilieren des Cobol-Programms

Selbst-Test

- **Das JCL Script in Abb. 3.2 übersetzt das Cobol Programm und speichert den Code. Wo ?**
- **Es führt den Code aber nicht aus. Warum nicht ?**

Um das gerade compilierte Cobol-Programm ausführen zu können, ist ein weiterer Member RUN erforderlich (Abbildung 3.3).

Aufgabe: *Erstellen Sie im Dataset <Ihre User-ID>.VSAM.CNTL den Member RUN . Passen Sie ihn wieder an Ihre Bedürfnisse an.*

Aufgabe: *Schreiben Sie in den Member RUN die Matrikel-Nummer hinein, nach der im VSAM-Dataset gesucht werden soll (Abbildung 3.3, hellgrün umrahmte Zahl). Starten Sie die Programmausführung mittels SUB.*

Der „STUDDS“-DD-Entry stellt die Verbindung für das COBOL-Programm dar. In Zeile 001400 in Abb. 3.1 wird festgelegt, wo die Daten abgelegt sind.

Über SYSIN übergeben wir die Matrikelnummer 1207864 des Records, der gesucht werden soll.

```
//PRAKT20R JOB (),NOTIFY=&SYSUID
//*
//* AUSFUEHREN DES COBOL-PROGRAMMS ZUM VSAM-ZUGRIFF
//*
//RUN      EXEC PGM=STUD
//STEPLIB  DD DSN=PRAKT20.VSAM.LOAD,DISP=SHR
//STUDDS   DD DSN=PRAKT20.VSAM.STUDENT,DISP=SHR
//SYSIN    DD *
1207864
/*
```

Abbildung 3.3: Member RUN

Nach dem submit steht nun im Job Log der Eintrag des Studenten (siehe Abbildung 2.2) mit allen zu ihm vorhandenen Daten (Abbildung 3.4).

```
SUCHE 1207864
MATNR:    1207864
STUDSERV: mai03wer
NACHNAME: Friedrich
VORNAME:  Boris
```

Abbildung 3.4 : Auszug aus dem Job Log

Wie können wir uns nun die in der Abbildung 3.4 gezeigte Ausgabe unseres Cobol-Programms anschauen? Dafür starten wir SDSF und rufen den Job Status auf (ST).

Nachdem der Job gefunden wurde, schauen wir uns entweder das Job Log durch das Kommando "S" direkt an oder wir geben statt "S" ein "?" ein.

Nun werden alle Ausgaben nach DD Cards gruppiert angezeigt. Wir interessieren uns nur für SYSOUT im Step RUN und schreiben vor diese Zeile ein „S“, um den Inhalt anzuzeigen.

Weitere Details dazu können Sie unserem Tutorial "Fehlersuche in OS/390 und OS/390-Anwendungen" entnehmen.

Aufgabe: ***Sehen Sie sich das Suchergebnis wie oben beschrieben an.***

Aufgabe: ***Suchen Sie nach Ihrer eigenen Matrikel-Nummer. Sehen Sie sich auch dieses Suchergebnis an. Fertigen Sie von diesem einen Screenshot entsprechend der Abbildung 3.5 an; dieser muss mindestens die rot umrandeten Zeilen enthalten. Der Screenshot soll im JPG-Format erstellt werden und darf 90 KByte nicht überschreiten.***

Aufgabe: ***Schicken Sie diesen sowie einen später anzufertigenden zweiten Screenshot unverpackt per E-Mail an Ihren Tutor, und zwar genau eine Abgabe-E-Mail mit je einem Anhang pro Screenshot.***

```
  Display  Filter  View  Print  Options  Help
 -------------------------------------------------------------------------------
 SDSF OUTPUT DISPLAY PRAKT20R JOB04593  DSID     4 LINE 9       COLUMNS 02- 81
 COMMAND INPUT ===>                                            SCROLL ===> 1
IEF142I PRAKT20R RUN - STEP WAS EXECUTED - COND CODE 0000
IGD104I PRAKT20.VSAM.LOAD                          RETAINED,  DDNAME=STEPLIB
IGD104I PRAKT20.VSAM.STUDENT                       RETAINED,  DDNAME=STUDDS
IEF285I   PRAKT20.PRAKT20R.JOB04593.D0000101.?         SYSIN
IEF285I   PRAKT20.PRAKT20R.JOB04593.D0000102.?         SYSIN
IEF373I STEP/RUN     /START 2006017.1928
IEF374I STEP/RUN     /STOP  2006017.1928 CPU    0MIN 00.03SEC SRB    0MIN 00.00S
IEF375I  JOB/PRAKT20R/START 2006017.1928
IEF376I  JOB/PRAKT20R/STOP  2006017.1928 CPU    0MIN 00.03SEC SRB    0MIN 00.00S
SUCHE 1187579
MATNR:    1187579
STUDSERV: mai91jhz
NACHNAME: Michaelsen
VORNAME:  Nils
******************************** BOTTOM OF DATA ********************************


  F1=HELP      F2=SPLIT     F3=END       F4=RETURN    F5=IFIND     F6=BOOK
  F7=UP        F8=DOWN      F9=SWAP     F10=LEFT     F11=RIGHT    F12=RETRIEVE
```

Abbildung 3.5: Ergebnis der Suche nach dem eigenen Namen

Selbst-Test

- **Unser Cobol Programm stellt das Verarbeitungsergebnis lediglich wohin ?**
- **Ginge es auch anders ? Wie?**
- **Haben Sie einen Vorschlag, wie man das Ganze benutzerfreundlicher gestalten könnte ?**

Die normale Ein- und Ausgabe würde darin bestehen, ein Cobol Anwendungsprogramm zu schreiben , welches auf den VSAM Dataset zugreift, ihn bearbeitet, abfragt und/Oder modifiziert.

Ein derartiges Beispiel ist verfügbar unter http://www.mainframes360.com/2009/04/vsam-tutorial-links.html.

oder

http://www.simotime.com/cblcbl01.htm#Introduction

Weitere Tutorials

https://sites.google.com/site/cobolmaterial/vasam-tutoriall

http://www.angelfire.com/folk/anoop/Vsamcob.pdf

4. Einführung in einen alternativen Index

4.1 AIX Cluster

Ein wichtiges Feld in jedem logischen Record ist der Key. Sein Inhalt kann zum Abrufen eines spezifischen logischen Records verwendet werden. Beispiele für Keys sind z.B. Personalnummern in einer Mitarbeiterdatei oder Teilenummern in einer Stücklistendatei. Ein KSDS muss mindestens einen Primärkey verwenden; er kann zusätzliche Sekundärkeys benutzen.

In einer VSAM Key Sequenced Organisation (KSDS) muss ein Record über einen eindeutigen Primärkey fester Länge in der gleichen Position innerhalb jedes logischen Records verfügen. Primärkeys können eine Größe von mindestens einem Byte und höchstens 255 Bytes haben.

Es können mehrere weitere Keyfelder in dem gleichen logischen Record vorhanden sein. Diese werden als alternate oder sekundäre Keys bezeichnet. Im Gegensatz zu den Primärkeys, die eindeutig und identisch sein müssen, können Sekundärkeys mit einem identischen Wert in mehr als einem logischen Record auftreten.

Ein VSAM Cluster, dessen Index Component Primärkeys enthält, wird als Basis Cluster bezeichnet. Jeder Sekundärkey erfordert einen weiteren VSAM Cluster, der als „Alternate Index“ Cluster (AIX Cluster) bezeichnet wird. Eine „Sphere“ ist ein Basis VSAM-Cluster mit seinen dazugehörigen AIX Clustern. Diese damit verbundenen associated Cluster sind die alternate Indizes (AIXs) des Basis Cluster.

Der Begriff AIX ist bei der Firma IBM mehrfach belegt. Neben VSAM bezeichnet der PowerPC sein weit verbreitetes Unix Betriebssystem ebenfalls als AIX (Advanced Interactive eXecutive). Wir verwenden AIX lediglich im Zusammenhang mit VSAM als Abkürzung für Alternate Index.

Ein AIX Cluster ist ein getrennter KSDS Cluster mit einer Index-Component und einer Daten Component. Die Index Component des AIX Clusters speichert die Sekundärkeys. Die AIX-Data Component speichert keine Daten, sondern Records, die Sekundärkeys und zugeordnete Primärkeys enthalten (Zeiger auf Daten in dem Basis-Cluster mit dem gleichen sekundären Keywert). Da identische Sekundärkeys in mehr als einem logischen Record auftreten können, kann jedem Sekundärkey in der AIX Data Component eine Gruppe von Primärkeys zugeordnet sein. Die Primärkeys in dem logischen AIX Record mit dem gleichen sekundären Keywert sind in aufsteigender Reihenfolge angeordnet.

Die Primärkeys in der AIX Data Component können nun benutzt werden, um über die Index Component des VSAM Basis Clusters auf den entsprechenden logischen Record zuzugreifen.

Jedes Feld in dem Basis Cluster Record kann als Sekundärkey verwendet werden. Der gleiche Basis-Cluster kann mehrere AIX Cluster mit unterschiedlichen Sekundärkey unterhalten. Es können mehrere Primärkey Records einen identischen Sekundärkey Wert benutzen. Beispiel: Der Primärkey ist die Mitarbeiternummer, und der Sekundärkey ist der Name der Abteilung. Offensichtlich kann die gleiche Abteilung mehrere Mitarbeiter haben.

Das Access Methods Services (AMS) utility program IDCAMS ermöglicht Definiton und Erstellung von AIXs. Es kann mit dem BLDINDEX Befehl aufgerufen werden. Ein AIX kann nur nach Definition des dazugehörigen Basis-Cluster definiert werden. Der BLDINDEX Befehl bewirkt einen sequentiellen Scan des angegebenen Basis Cluster. Während des Scans werden Sekundär Keywerte und Primärkeys extrahiert. Hiermit werden die AIX Records in der Data Component des AIX Clusters erzeugt.

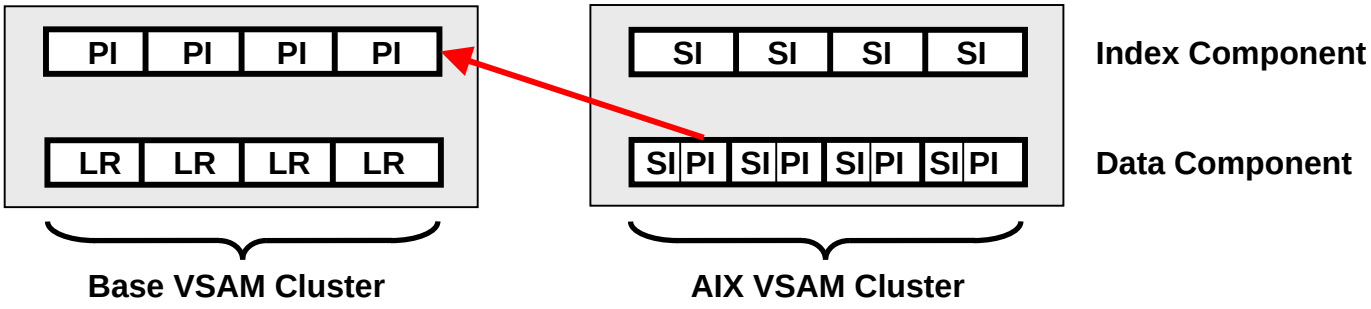


Abbildung 4.1

LR	**Logical VSAM Data Record**
PI	**Primärindex**
SI	**Sekundärindex**

4.2 Alternate Index Pfade

Vor einem Zugriff auf KSDS über eine alternate Index muss ein Pfad (Path) definiert werden. Mittels eines Pfades kann auf einen Basis-Cluster über die alternate Indizes zugegriffen werden. Ein Pfad wird mittels des AMS DEFINE PATH Befehls definiert und benannt. Mindestens ein Pfad muss für jede der alternaten Indizes definiert werden.

Der Pfadname verweist auf das Basis-Cluster und AIX-Cluster Paar. Wenn ein Anwendungsprogramm einen OPEN Befehl ausführt, werden sowohl der Base Cluster als auch der AIX Cluster geöffnet (opened).

5. Erweiterung um alternativen Index

Um nun auch über das Studserv-Kennung auf einen Record zugreifen zu können benötigt man einen alternativen Index Cluster. Der alternative Index Cluster hat seine eigene Kopie der Records und wird über einen PATH mit dem Base VSAM Cluster verbunden.

Einen alternativen Index Cluster kann man wieder mit einem geeigneten JCL-Script definieren. Es folgt geeigneter Programmcode.

```
//PRAKT20D JOB (),NOTIFY=&SYSUID
//*
//* DEFINE VSAM ALTERNATE INDEX
//*
//DEFCLS   EXEC PGM=IDCAMS,REGION=4096K
//SYSPRINT DD SYSOUT=A
//SYSIN    DD *

  DEFINE ALTERNATEINDEX ( -
           NAME(PRAKT20.VSAM.STUDENT.ALTINDEX) -
           RELATE(PRAKT20.VSAM.STUDENT) -
           VOL(SMS009) -
           RECORDSIZE(55 55) -
           KILOBYTES(100 100) -
           KEYS(8 7) -
           UPGRADE )

  DEFINE PATH ( -
           NAME(PRAKT20.VSAM.STUDENT.PATH) -
           PATHENTRY(PRAKT20.VSAM.STUDENT.ALTINDEX) )

  BLDINDEX INDATASET(PRAKT20.VSAM.STUDENT) -
           OUTDATASET(PRAKT20.VSAM.STUDENT.ALTINDEX) -
           SORTCALL
/*
```

JCL Script zur Erstellung eines Alternate Index Cluster
Abbildung 5.1

Dieses JCL Script übernimmt verschiedene Aufgaben. Zuerst wird der alternative Index angelegt. Der alternate Key hat eine Länge von 8 Zeichen und beginnt bei Offset 7.
Damit der alternative Index funktioniert, muss ein PATH angelegt werden.
Zuletzt muss der alternative Index erstmalig erstellt werden. Das Wiederholen dieses Schrittes ist durch die Angabe von UPGRADE beim Definieren des alternativen Indexes nicht nötig.

Selbst-Test

- **Wodurch unterscheidet sich der alternative Index von dem primary Index?**
- **Was ist die Aufgabe des DEFINE PATH Kommandos ?**
- **Was ist die Aufgabe des BLDINDEX Kommandos ?**

Aufgabe: *Legen Sie ein entsprechendes JCL-Script als Member DEFAIX im PDS „PRAKT20.VSAM.CNTL“ an.*

Aufgabe: *Führen Sie dieses JCL-Script aus und kontrollieren Sie, ob der alternative Index tatsächlich angelegt wurde. Der VSAM-Cluster müsste unter anderem um weitere Komponenten PRAK<xxx>.VSAM.STUDENT.ALTINDEX.* erweitert worden sein.*

Jetzt brauchen wir ein Cobol Programm, welches den Alternate Index benutzt. Hierzu kopieren wir das ursprüngliche Cobol Programm (welches den Primärindex benutzte in einen neuen Member. Wir ändern es dann so ab, dass es den Alternate Index benutzt.

Aufgabe: *Kopieren Sie den Cobol-Quelltext in einen neuen Member STUD2.*

Aufgabe: *Modifizieren Sie den Member STUD2 entsprechend der nachfolgenden Liste.*

Hier die Liste der Änderungen in STUD2 (- heißt löschen, + heißt hinzufügen):

```
            ORGANIZATION IS INDEXED
            ACCESS IS DYNAMIC
            RECORD KEY IS MATNR
+           ALTERNATE RECORD KEY IS STUDSERV
            FILE STATUS IS FSTAT-CODE VSAM-CODE.
-          SELECT MATNR-IN
+          SELECT STUDSERV-IN
            ASSIGN TO SYSIN
            ORGANIZATION IS SEQUENTIAL
-           FILE STATUS IS MATNR-IN-CODE.
+           FILE STATUS IS STUDSERV-IN-CODE.
       DATA DIVISION.
       -------------
       FILE SECTION.
@@ -30,18 +31,18 @@
           05 STUDSERV    PIC X(8).
           05 NNAME       PIC X(20).
           05 VNAME       PIC X(20).
-      FD MATNR-IN
+      FD STUDSERV-IN
           RECORDING MODE F
           BLOCK 0 RECORDS
           RECORD 80 CHARACTERS
           LABEL RECORD STANDARD.
-        01 MATNR-RECORD  PIC X(80).
+        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 MATNR-IN-CODE  PIC X(2).
+        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.
@@ -51,22 +52,22 @@
         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).
+      01 WS-STUDSERV-IN-RECORD.
+        05 WS-STUDSERV-IN  PIC X(8).
+        05 FILLER          PIC X(72).

       PROCEDURE DIVISION.
-          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.
           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.
+          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
@@ -77,7 +78,7 @@
           MOVE NNAME TO WS-NNAME.
           MOVE VNAME TO WS-VNAME.
           CLOSE STUD-FILE.
-          CLOSE MATNR-IN.
+          CLOSE STUDSERV-IN.
           DISPLAY "MATNR:    " WS-MATNR.
           DISPLAY "STUDSERV: " WS-STUDSERV.
           DISPLAY "NACHNAME: " WS-NNAME.
```

Die Liste der Änderungen im Cobol-Progamm STUD sollte helfen, alle Änderungen einzubauen.

Es wird nun ein JCL-Script als Hilfsmittel benötigt, um das neue COBOL-Programm zu compilieren.

6. Arbeiten mit dem alternativen Index

Aufgabe: *Kopieren Sie den Member COMPILE im PDS „PRAKT20.VSAM.CNTL“ nach COMP2 und passen Sie diesen so an, dass Sie ihn zum Compilieren des neuen Cobol-Programms einsetzen können. Diese Anpassung muss bewirken, dass der alte ausführbare Member "STUD" in "PRAKT20.VSAM.LOAD" nicht überschrieben wird. Statt dessen soll hier ein zweiter Member "STUD2" angelegt werden.*

Aufgabe: *Compilieren Sie das neue Cobol-Programm.*

Nun bleibt uns noch, das gerade compilierte neue Cobol-Programm auszuführen. Dazu kopieren wir den Member RUN nach RUN2 und passen ihn an. Bei einem Vergleich von RUN mit RUN2 (Abbildung 7) fällt auf: In RUN2 muss noch zusätzlich etwas eingefügt werden, um den alternativen Index bekannt zu machen (über den erstellten PATH).

Nach Ausführung findet man die Daten des Studenten wie beim vorigen Cobol-Programm im Job Log des Jobs.

Aufgabe: *Erstellen Sie einen letzten neuen Member RUN2 in PRAKT20.VSAM.CNTL (siehe Abbildung 6.1) und führen Sie mit dessen Hilfe das zweite Cobol-Programm zu Test-Zwecken mehrfach aus.*

Aufgabe: *Suchen Sie wieder nach Ihrem eigenen Datensatz, diesmal über Ihr eigenes Studserv-Kennung. Fertigen Sie nach dem Muster von Abbildung 6.2 einen zweiten Screenshot an. Auch dieser soll im JPG-Format erstellt werden und darf 90 KByte nicht überschreiten und muss mindestens die rot umrandeten Zeilen enthalten.*

Aufgabe: *Schicken Sie nun beide Screenshots unverpackt und im JPG-Format an Ihren Tutor, und zwar in genau einer E-Mail mit je einem Anhang pro Screenshot.*

```
//PRAKT20R JOB (),NOTIFY=&SYSUID
//*
//* AUSFUEHREN DES COBOL-PROGRAMMS ZUM VSAM-ZUGRIFF
//* UEBER ALTERNATIVEN INDEX (STUDSERV-KUERZEL)
//*
//RUN      EXEC PGM=STUD2
//STEPLIB  DD DSN=PRAKT20.VSAM.LOAD,DISP=SHR
//STUDDS   DD DSN=PRAKT20.VSAM.STUDENT,DISP=SHR
//STUDDS1  DD DSN=PRAKT20.VSAM.STUDENT.PATH,DISP=SHR
//SYSIN    DD *
mai00aaa
/*
```

Abbildung 6.1: JCL-Script zum Ausführen des zweiten Cobol-Programms

```
  Display  Filter  View  Print  Options  Help
 ------------------------------------------------------------------------------
 SDSF OUTPUT DISPLAY PRAKT20R JOB04622  DSID     4 LINE 10      COLUMNS 02- 81
 COMMAND INPUT ===>                                           SCROLL ===> 1
IEF142I PRAKT20R RUN - STEP WAS EXECUTED - COND CODE 0000
IGD104I PRAKT20.VSAM.LOAD                          RETAINED,  DDNAME=STEPLIB
IGD104I PRAKT20.VSAM.STUDENT                       RETAINED,  DDNAME=STUDDS
IGD104I PRAKT20.VSAM.STUDENT.PATH                  RETAINED,  DDNAME=STUDDS1
IEF285I   PRAKT20.PRAKT20R.JOB04622.D0000101.?         SYSIN
IEF285I   PRAKT20.PRAKT20R.JOB04622.D0000102.?         SYSIN
IEF373I STEP/RUN     /START 2006018.1558
IEF374I STEP/RUN     /STOP  2006018.1558 CPU    0MIN 00.04SEC SRB    0MIN 00.00S
IEF375I  JOB/PRAKT20R/START 2006018.1558
IEF376I  JOB/PRAKT20R/STOP  2006018.1558 CPU    0MIN 00.04SEC SRB    0MIN 00.00S
SUCHE mai91jhz
MATNR:    1187579
STUDSERV: mai91jhz
NACHNAME: Michaelsen
VORNAME:  Nils
****************************** BOTTOM OF DATA ********************************

  F1=HELP      F2=SPLIT     F3=END       F4=RETURN    F5=IFIND     F6=BOOK
F7=UP        F8=DOWN      F9=SWAP     F10=LEFT     F11=RIGHT    F12=RETRIEVE
```

Abbildung 6.2: Ergebnis der Suche über das Studserv-Kennung

7. Tips

Tip Nr. 1

Die Matrikel-Nummern im sequentiellen Dataset müssen ausnahmslos aufsteigend geordnet sein. Wenn man das nicht beachtet, entstehen sehr schwer auffindbare Fehler.

Tip Nr. 2

Problem, dass der alternative Index, z.B. mam07abc, nicht gefunden wird:

Beim Durchsuchen der alternativen Indizes wird auf Groß- und Kleinschreibung geachtet. Möglicherweise ist in Ihrem ISPF-Editor eine Option aktiviert, die korrekt eingetippte klein geschriebene alternative Indizes fälschlicherweise in Großbuchstaben konvertiert. Wie sich dieses Problem beheben lässt, ist auch in diesem Forum in einem anderen Thread beschrieben.

Im Tutorial wurde als Aufgabe unter anderem gestellt, einen sequentiellen Dataset anzulegen und später dann auch zu editieren. Wie man das im Detail tun kann, ist im Tutorial nicht beschrieben. Doch denke ich, dass dies eine einfache Aufgabe für Sie ist, dies selbst herauszufinden.

7. Anhang

Für die Durchführung des Tutorials werden die folgenden Resourcen benötigt:

Inhalt der Datei VSAMcode.zip

CLEAN	**Delete VSAM Cluster**
COMPILE	**Compilieren des Cobol Programms, P. 8**
DEFAIX	**Define alternate Index, p.10**
DEFCLUST	**Erstellen VSAM Cluster, p.3**
STUD	**Cobol Program, P. 7**

Die Datei VSAMcode.zip enthält diese Resources als zip Archiv. Sie kann heruntergeladen werden unter

www.cedix.de/Vorles/Band3/Resources/VSAMcode.zip