Table of Contents

Introduction to VSAM

Virtual Storage Access Method (VSAM) is two things. It is a type of disk data set, and an Application Programming Interface (API) for using these data sets. Or in other words, an access method, such as sequential access, or direct access.

- VSAM data sets must be stored on disk. Tape data sets cannot be VSAM.
- VSAM is included free with z/OS, z/VSE, and z/VM, though there can be differences between these operating systems.

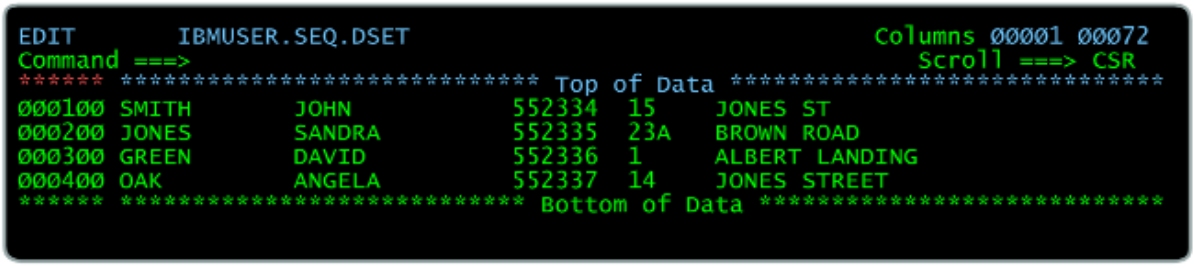
VSAM and Other Access Methods

There are several different data set types on the mainframe, each with advantages and disadvantages.

VSAM data sets are usually used by programs to store and manage data. They cannot be directly browsed or edited from ISPF without special software like IBM File Manager for z/OS or Compuware File-AID.

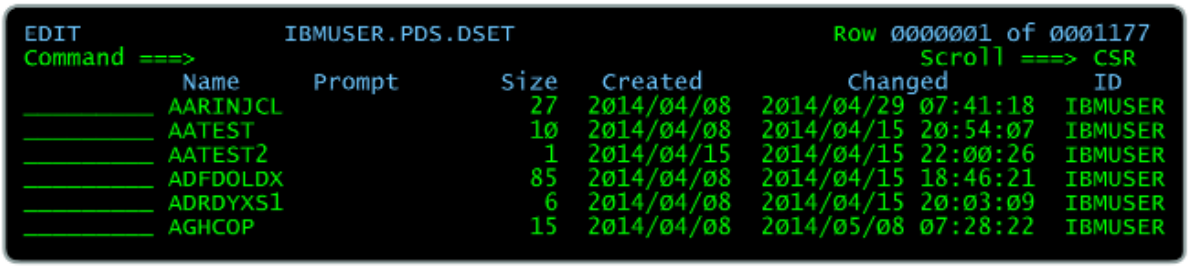
Sequential

- Data sets you can edit or browse in ISPF
- It is ideal for text
- Access method: QSAM or BSAM



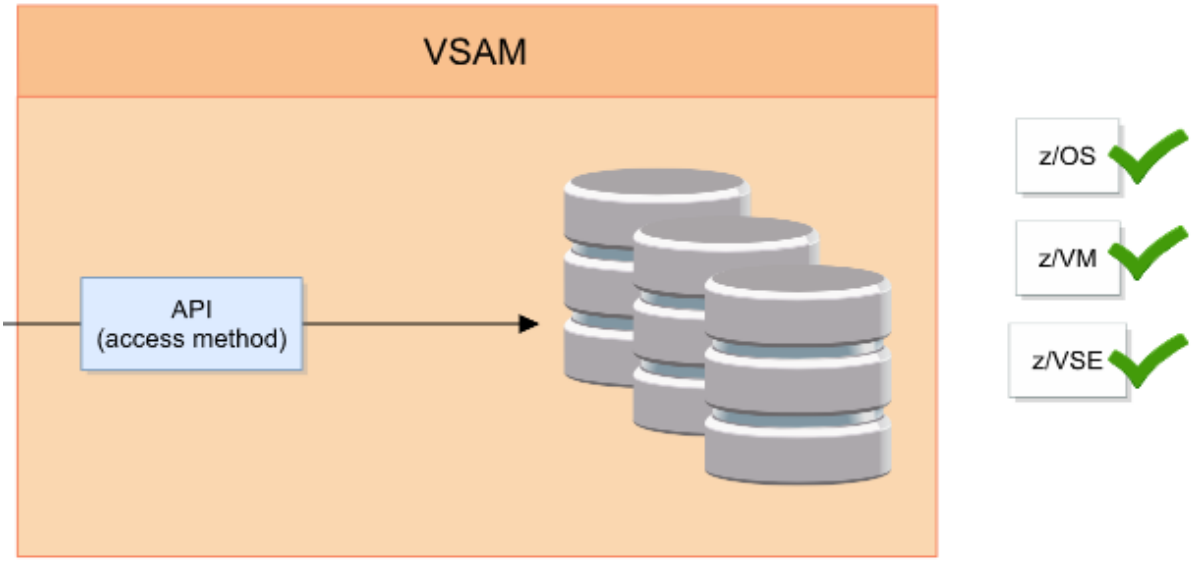
Partitioned

- A data set that has been divided into partitions called members.
- Each member contains sequential information such as data records, JCL statements or program code
- Access method: BPAM



VSAM

- One or more connected data sets
- It is ideal for storing data



VSAM Features

VSAM provides extra features that make it excellent for storing data. These include the following:

- **Speed** - VSAM data sets are the fastest data sets available for most data-related needs. They are excellent for heavy workloads.
- **Size** - VSAM data sets can be quite big; with the introduction of extended addressability, a DB2 VSAM data set, for example, could yield a maximum size of128TB.
- **Extents** - VSAM data sets can have up to 123 extents on each disk, up to a maximum of 255 extents. They can have even more if configured by the systems admin.
- **Access** - VSAM data sets can be accessed in three ways:
    - Sequential access - one record at a time from the beginning of the data set
    - Direct access - using a key or location to directly access a record
    - Skip-sequential access - locating on record directly and then processing later records sequentially from this record

VSAM Extended

VSAM data sets have been used since the 1970s. More recently, IBM introduced a newer format of VSAM data set; VSAM extended (Extended Format VSAM).

Extended Format VSAM is a VSAM data set with extended features that must be managed by DFSMS. Application programs see no difference between VSAM and Extended Format VSAM. However Extended Format VSAM provides some advanced features for the storage administrator.

- **Sharing** - Extended format VSAM data set records can be shared by many different users at the same time, even from different systems. This is called record level sharing (RLS).
- **Size** - Prior to DFSMS 1.6, a 4 gigabyte architectural limit for data set size was imposed. With the introduction of extended addressability and extended format VSAM data sets, this maximum size was increased to 128 terabytes.
- **Compression** - Extended format VSAM data sets can be stored in a compressed format on disk.
- **Speed** - Extended format VSAM provides features to improve VSAM performance, including intelligent buffering and data set striping.

VSAM Users

VSAM is used by many different software products, and even z/OS itself. Listed below are a few examples of this:

- **IMS** - Stores some IMS databases as VSAM data sets.
- **DB2** - Uses VSAM to store Db2 objects.
- **z/OS** - Many system data sets including catalogs and SMF data sets are VSAM.
- **CA ACF2 and RACF** - User information and security rules are stored in VSAM data sets.
- **Other z/OS File Types** - z/OS uses VSAM to implement other file types such as HFS, zFS and PDSE.

VSAM Data Sets

ESDS

There are five basic types of VSAM data set. The simplest is called the Entry Sequenced Data Set (ESDS). ESDS data sets are similar to normal QSAM data sets, and can hold fixed or variable length records in any order.

ESDS data sets can be accessed sequentially or directly. They cannot be accessed skip-sequentially.

ESDS data sets are excellent for applications such as logs where sequential access is needed.

How ESDS Data Sets Work

- **Structure** - ESDS data sets hold fixed or variable length records in any order.
- **Sequential access** - An ESDS record can be located sequentially by checking each record, starting from the beginning of the data set.
- **Direct access** - An ESDS record can be located directly if the location, known as the relative byte address (RBA) is known. The location could have been determined from a previous sequential search, or by remembering the location when the record was first inserted.
- **New Records** - They must be added at the end of the data set
- **Removing Records** - Records cannot be physically deleted from an ESDS. Records that are no longer needed can be marked as deleted, but will remain in the data set until the ESDS is completely rebuilt.
- **Updating Records** - Records can be updated where they are, providing the length remains the same. If the length changes, the record must be marked as deleted, and a new record added at the end.

Control Intervals

A VSAM control interval (CI) is the basic building block of a VSAM data set. It holds one or more VSAM records.

When a record in a VSAM data set is read, it must be moved into memory from a DASD device. Similarly, when a record is written, it is moved from memory to a DASD device.

- When a record is read from a VSAM data set on disk, the entire control interval is moved.

For VSAM data sets, the entire CI is moved, not just one record. This can speed up VSAM processing. The CI is similar to the block for sequential and partitioned data sets. The size of the control interval is defined by the user when creating the VSAM data set.

CIs and ESDS

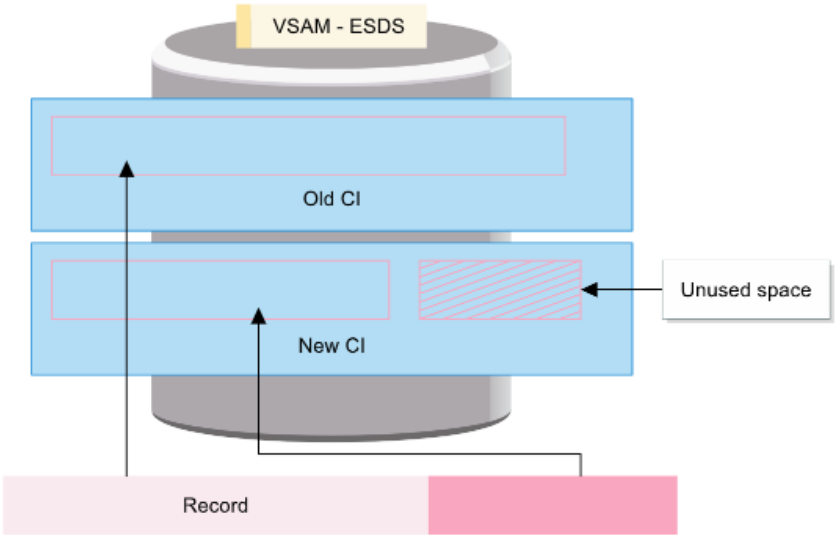
For ESDS data sets, each record is inserted at the end of the data set, at the end of the most recent CI. If there is not enough room at the end of the CI, a new one is started.

If a record is larger than the CI size, it can use more than one CI, providing the VSAM data set is defined as spanned.

Spanned records
If the record is bigger than the CI size, it cannot be stored.

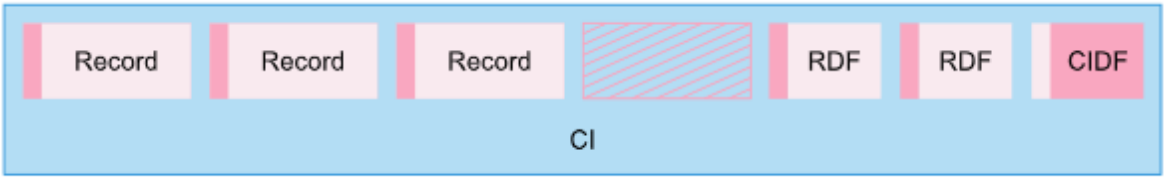
However if the VSAM data set is defined as spanned, the record can span across more than one CI. This is called a spanned record.

Spanned records occupy entire CIs. Any unused space in the CI cannot be used.



CI Structure

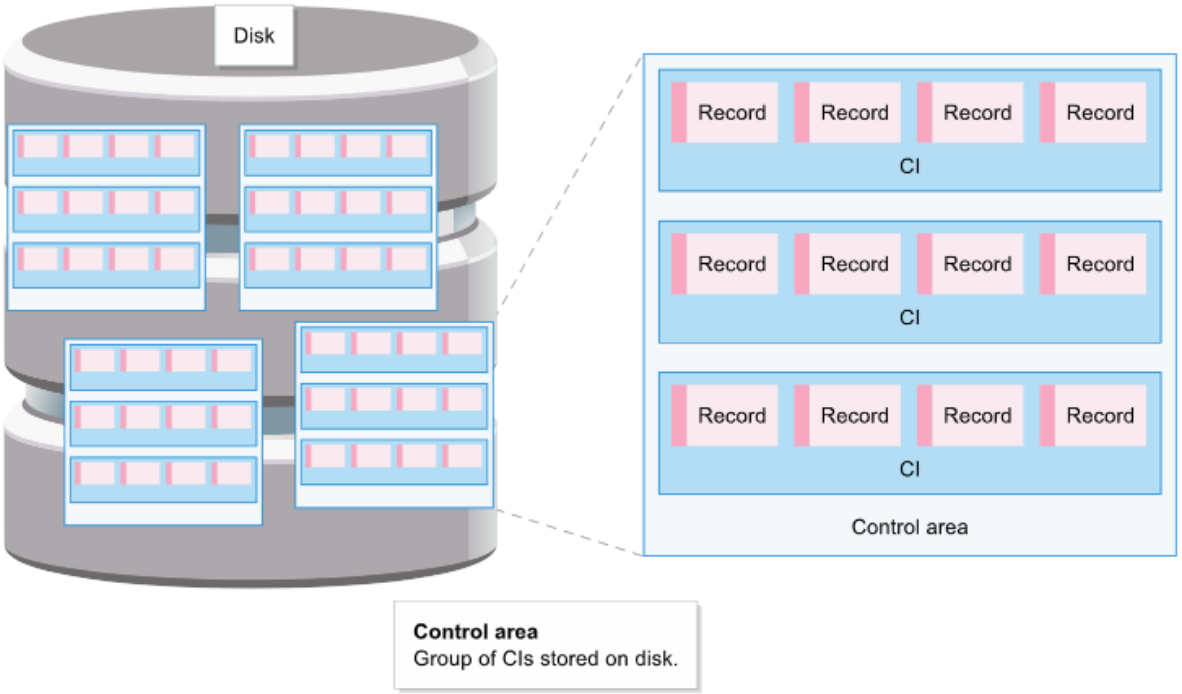
VSAM not only stores records in the CI. It also stores other information needed to manage the CI. This information includes the following



- **Records** - Data records are stored in each interval.
- **Unused Space** - For ESDS data sets, this cannot be used unless it is after the final record in the data set.
- **Record Definition Field (RDF)** - The RDF stores the length of each record. There is one per record for VSAM data sets with variable length records. VSAM data sets with fixed length records only need two; one for the record length and one recording the number of records.
- **Control Interval Definition Field (CIDF**) - This is one 4-byte field holding the size and location of unused space in the CI.

Control Areas

VSAM CIs are stored on disk in groups. These groups are called control areas (CAs). One VSAM data set can have one or more CAs.



The size of the CA is determined by the system when the data set is defined.

RRDS

The relative record data set (RRDS) is another type of VSAM data set.

How RRDS Data Sets Work

- **Structure:**
  - RRDS data sets consist of fixed length areas called slots. Slots are pre-formatted when the data set is created, or whenever a new CA is created.
  - Records are inserted and deleted into these slots.
  - They also have the same CI structure as ESDA data sets, including RDFs and CIDFs. Any unused space created as a result of CI fragmentation cannot be used to store data.
  - There is one RDF for every slot; each RDF is used to indicate whether the corresponding slot is being used or is empty.
- **Limitations:**
  - All records must be the same length.
  - No spanned records; records cannot span slots or CIs.

RRDS Access

RRDS data sets have some advantages over ESDS data sets:

- Records can be added and deleted within the data set.
- Records can be directly accessed by specifying the slot number. This is called a Relative Record Number, or RRN. The first slot has an RRN of 1.
- Skip-sequential processing can be used.

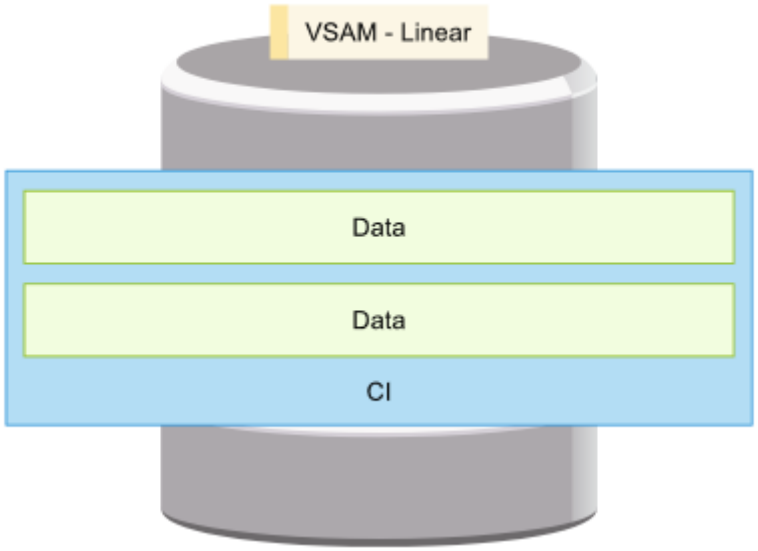
RRDS data sets are excellent for applications with fixed length records where direct access by relative record numbers is needed. In reality, RRDS data sets are rarely used today.

How to access:

- **Sequential access** - Like ESDS data sets, records can be located sequentially by checking each record, starting from the beginning of the data set.
- **Direct access** - A RRDS record can be located by specifying the RRN; for example to access the 8th record the RRN would be 8.
- **Skip-sequential access** - A RRDS record can be located directly by specifying the RRN, and subsequent records accessed sequentially.
- **New Records** - These can be inserted into any free slot. The application program can specify the destination slot; this is known as direct insertion. Or the application program can request the record be inserted in the next slot; this is known as sequential insertion.
- **Removing Records** - Any record can be deleted, and the slot reused.

Linear Data Sets

Linear data sets are different from other VSAM data sets. They do not hold records, but simply strings of data.



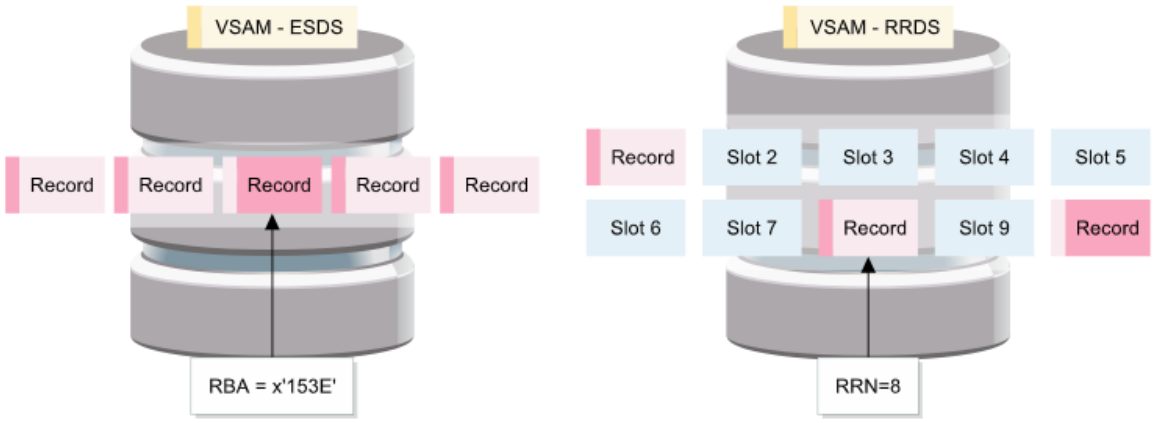
Linear data sets are usually used by programs and applications that want to specify their own internal record structure. Db2 uses linear data sets to store Db2 objects.

How Linear Data Sets Work

- **Structure** - Linear data sets hold only data. There are no records, and no RDFs or CIDFs. Programs accessing the data set must internally determine their own record structure.
- **Access** - Programs use z/OS data-in-virtual (DIV) or similar services to insert, update, read and delete data as if it was memory.

VSAM Indexes

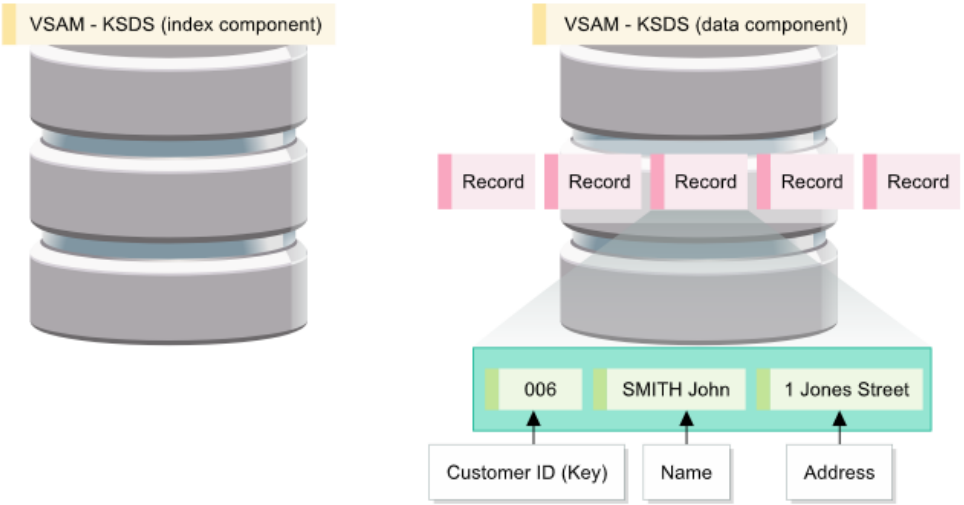
ESDS and RRDS VSAM data sets are excellent for sequential access. However, to directly locate a record inside these data sets a program must know either the RBA for ESDS, or the RRN for RRDS.



Although there are ways to obtain the RBA or RRN, it can be difficult. VSAM solves this problem with another VSAM data set type; the key sequenced data set (KSDS).

KSDS

A KSDS is actually two VSAM data sets that are used together. Each data set is called a component.



The data component holds the data in the same way as an ESDS; fixed or variable length. A part of each record is the record's key.

- This key can be any 1-255 contiguous bytes in the record that are unique.
- There cannot be two records with the same key.
- This key must also be the same length and position for all records.
- This key can be used to locate the record.

Record Insert into KSDS

Records in the data component of a KSDS are stored in key order.

They can be stored sequentially or directly.

Once a record has been inserted, its key cannot be altered. To change a record's key, the record must be deleted and re-inserted.

- **Free Space** - When defining VSAM data sets (other than RRDS), the amount of space to leave free in every CI is specified: the free space. RRDS data sets do not have free space.
- **Sequential insert** - KSDS records can be inserted sequentially. This often happens when the data set is first created and the data is loaded in. Records are inserted in key order until the free space limit is reached or the CI is full. Records are then inserted in the next CI.
- **Direct insert** - Records can also be inserted directly. This is the normal processing when using a VSAM data set that has already been loaded with data. A record is inserted in a space so that it is in key order. If there is not enough room to insert a record, other records are moved into the free space to make room for the new record.

Record Update in KSDS

Records in a KSDS data component can be updated and their length changed.

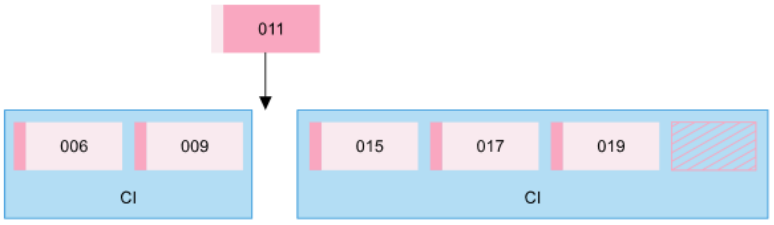
- **Same Length** - If a record is updated and the length is the same, the processing is the same as for an ESDS or KSDS.
- **Reduced Length** - If the length of the record being updated is reduced, then the remaining space is now free space. It can be reused as necessary for record inserts or updates.
- **Increased Length** - If the length of the record being updated is increased, then the records are moved into the free space to make room for the extra length.

CI Split

If there is not enough free space in a CI for a new record or increased length of an existing record, the CI is split into two. This is called a CI split.

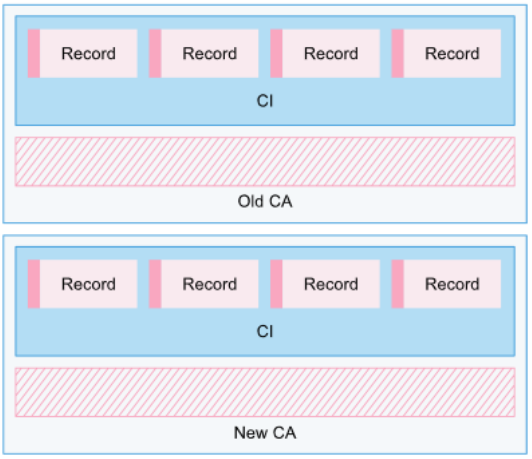
Suppose a new record is ready to be inserted, or an existing record will be updated and its length increased. However, there is not enough free space in the CI for the new data to fit.

- The CI is split into two.
- The second half is moved to a free CI in the CA
- There is now enough free space to insert the new record, or to expand an existing one.



CA Split

In a CI split, half of the CI is moved to a spare CI in the CA. Like CI, the amount of space to leave free in every CA can be specified when defining the VSAM data set - the free space.



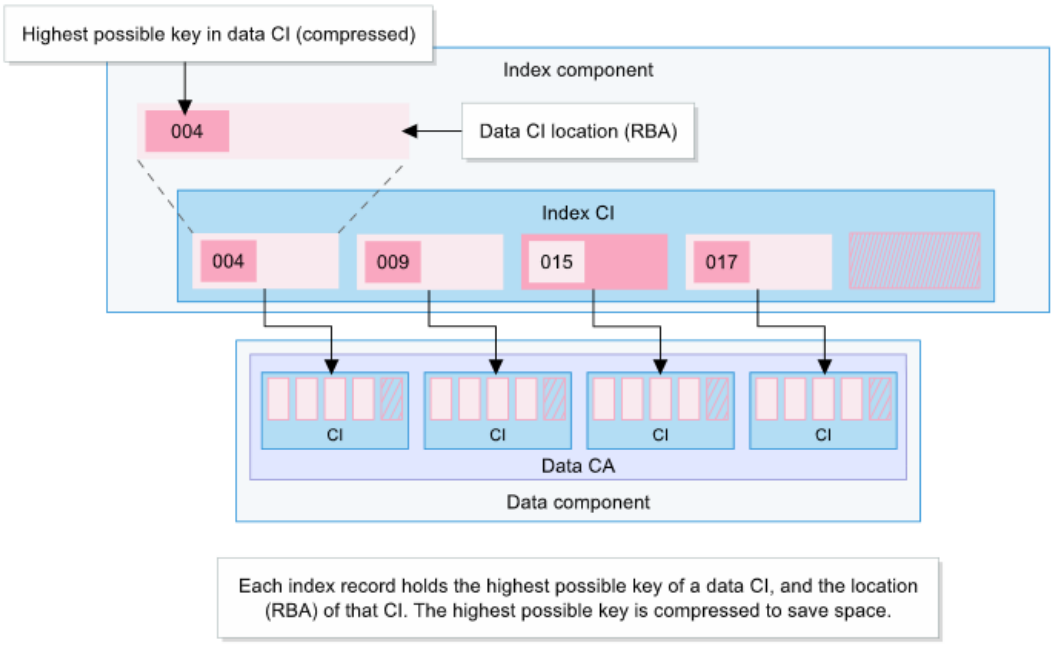
During sequential inserts, this free space is not used. If there is a CI split, one half of the CI is moved into a new CI in the CA free space.

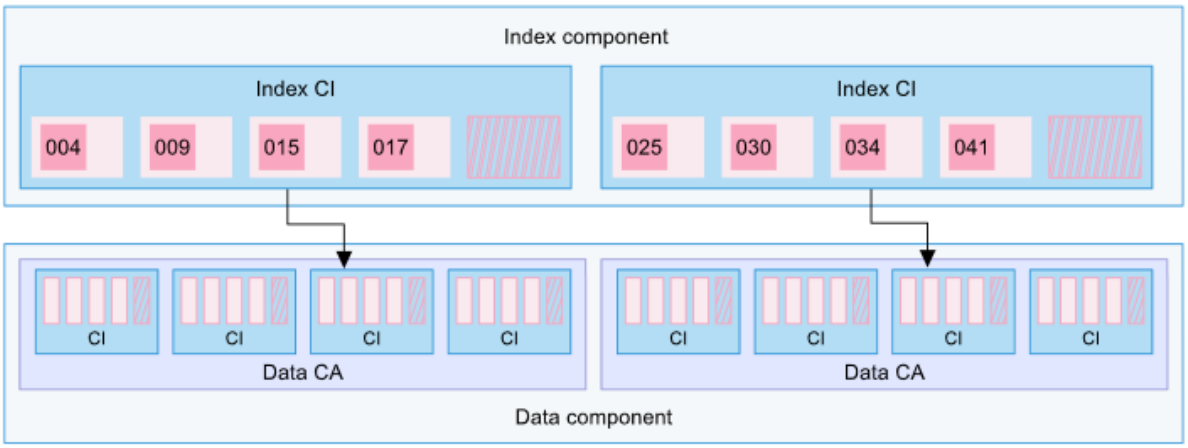
- If there is not enough space in the CA for another CI, then the CA is split into two.
- This is when one half of the cold CA is moved to a new CA at the end of the VSAM data set.
- The CI split can now be processed.

KSDS Index

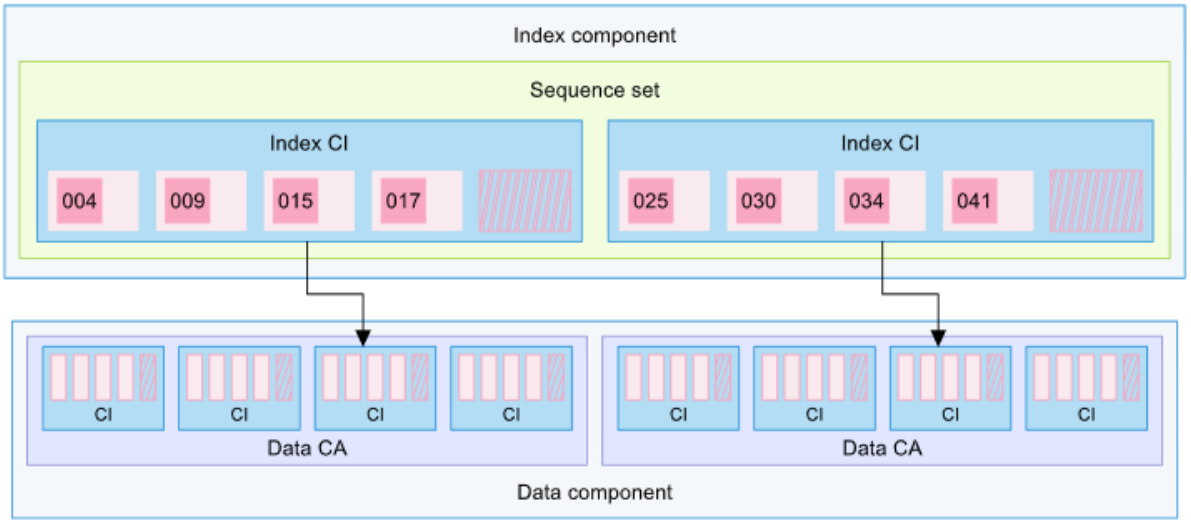
The index component of a KSDS holds exactly that, an index into the data component. This is sometimes called the prime index. It is built automatically as records are inserted, updated, or deleted. It is also updated automatically for any CA or CI splits.

This index allows VSAM to locate any data record given its key value.

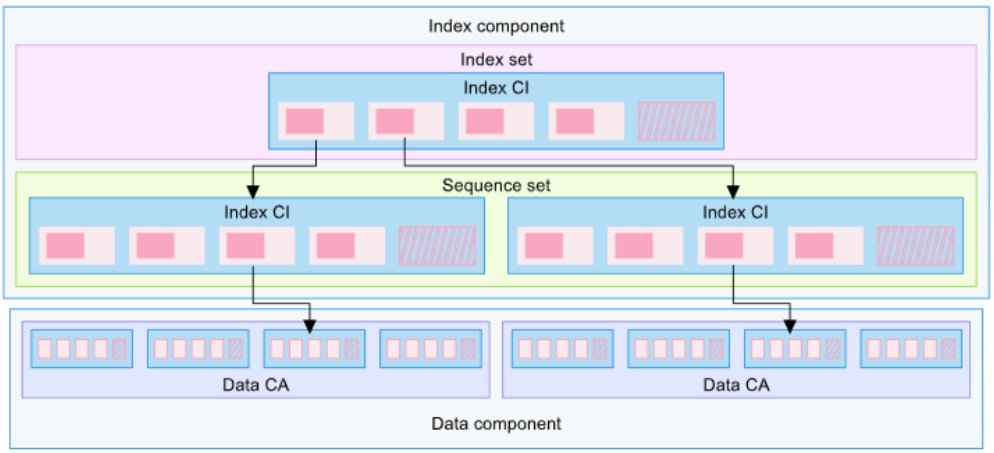




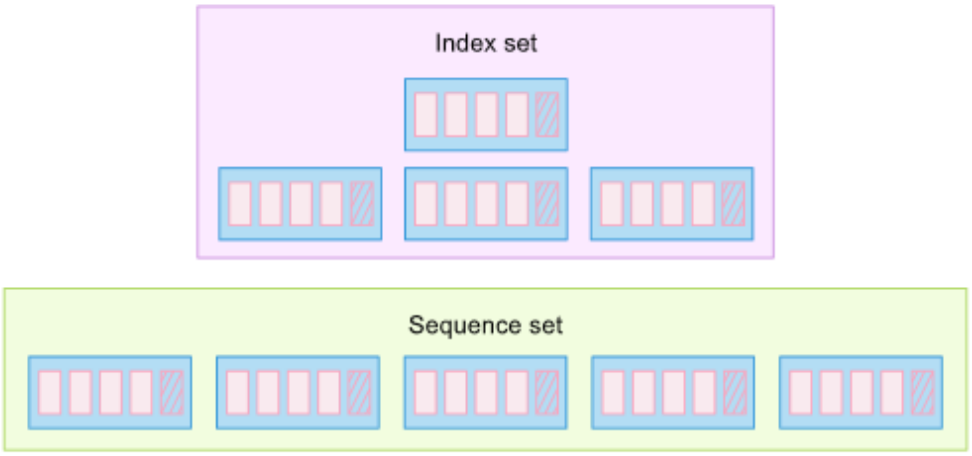
Every index CI holds all the indexes for one data CA.



This index is called the sequence set.



Larger VSAM data sets will have a second index level, where each record points to a sequence set CI. When accessing a VSAM data set, this second level is accessed first to find the sequence set index record. The sequence set record is then accessed to locate the data record.



VSAM data sets can have many index levels. However, there will always be only one sequence set.

Index Component Structure

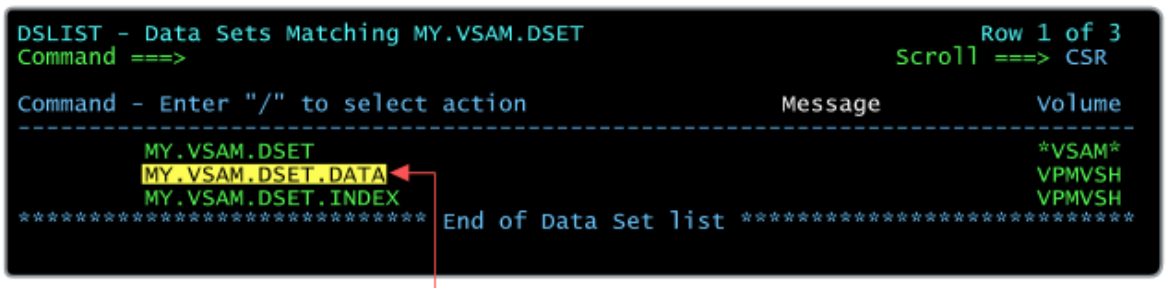
The KSDS index component is very similar in structure to the data component.

- **CIs and Free Space** - The index component is broken into CIs. The size of this CI can be determined when creating the VSAm data set. The free space of each index CI can also be specified when creating the VSAM data set.
- **CAs and Free Space** - Like the data component, the index CIs are grouped into CAs. Like all CAs, the size is determined by the system. The free space of each index CA can be specified when creating the VSAM data set.
- **CI Split** - Index CIs will split if there is not enough space in the CI for another index record.
- **CA Split** - If the index CA does not have enough space for a new index CI, a CA split will occur.

KSDS Data Set Names

This screen shows the ISPF DSLIST (option 3.4) listing of a VSAM KSDS. All these components together are called a VSAM cluster.

The name of each component is specified when the VSAM data set is created, and can be any valid data set name. However, most users use the cluster name for the data and index components, ending each with something like DATA or INDEX respectively.



The Components:

- **Cluster Component** - A component that relates the data and index components together. The cluster component is not a data set, only a catalog entry. All VSAM data sets have a cluster component, and any program or utility accessing a VSAM data set refers to the cluster component.
- **Data Component** - Every VSAM data set has a data component
- **Index Component** - Only KSDS and VRRDS data sets have an index component

KSDS Processing

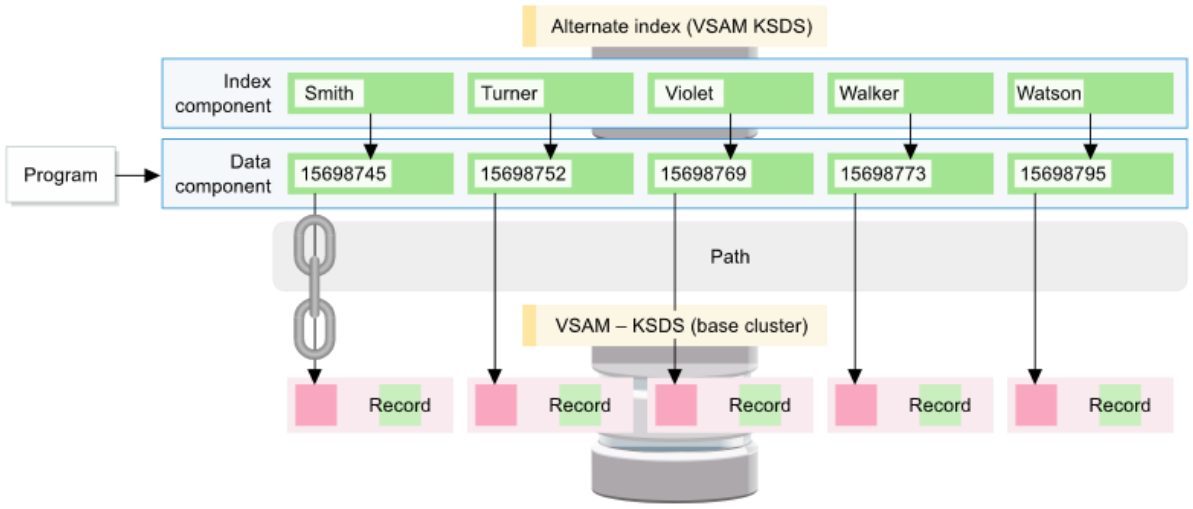
KSDS data sets can be accessed sequentially, directly, or skip-sequentially.

- **Sequential access** - Like ESDS and RRDS data sets, records can be located or processed sequentially by accessing each record. Records are accessed from the lowest key to the highest key. Records in a KSDS are not necessarily next to each other, particularly if a CI or CA split has occurred. So sequential processing locates each record using the sequence set in the index component.
- **Direct access** - Records in a KSDS can be located directly by specifying the record key. The KSDS index is accessed to locate the record with this corresponding key.
- **Skip-sequential access** - A record in a KSDS can be located directly by specifying the record key. Records after this record are processed sequentially.

Alternate Indexes

Suppose a VSAM record can be accessed by two different keys. For example, a telephone directory may have a unique customer number and telephone number.

VSAM allows this using an alternate index. An alternate index is the way of accessing a VSAM data set using a key that is different from the defined key. So in the example, the VSAM key could be the telephone number and the alternate index is the customer number.



Alternate indexes can also be used to add an index to an existing ESDS data set.

- An alternate index is a group of pointers into a VSAM KSDS or ESDS data set using a different key. The VSAM data set holding the data is called the base cluster.
- For example, a KSDS may use an eight byte customer ID at the beginning of the record as the primary key. An alternate index could be defined to use the 20 byte customer name beginning at position 8 in the record.

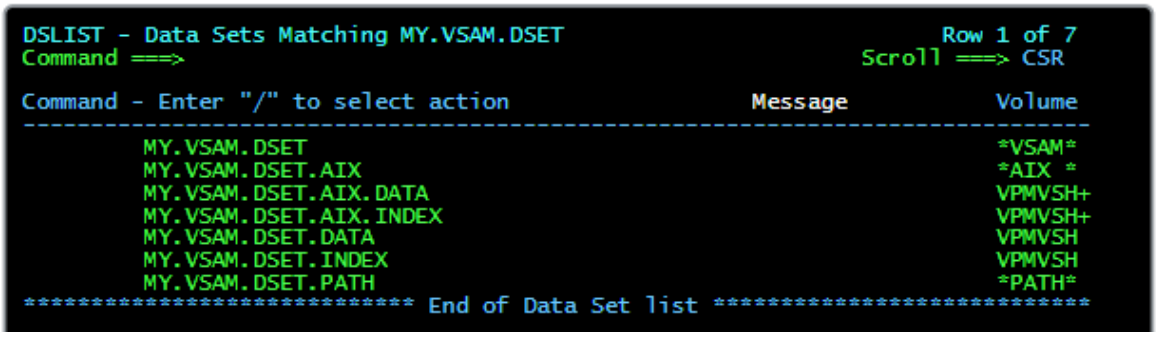
- One base cluster can have any number of alternate indexes.
- There is one alternate index object for every alternate key in a base cluster. Like the primary key of the base cluster, alternate keys can be any 1-255 contiguous bytes in a VSAM record. However, unlike the primary key, alternate keys do not have to be unique. There can be more than one base cluster record with the same alternate key.
- The alternate index object is itself a VSAM KSDS that is created after the base cluster.
    - The data component of the alternate index holds pointers to the base cluster, either an RBA address (if the base cluster is an ESDS) or the prime key (if the data component is a KSDS).
    - The index component works the same as the index component of a VSAM KSDS.
- Once the alternate index has been built, an object called a path is created to link the alternate index to the base cluster.
- Programs access the path as if it was the base cluster, using the alternate key. The alternate index can be automatically updated by VSAM whenever the base cluster changes.

VSAM Sphere

This screen shows the ISPF DSLIST (option 3.4) listing of a VSAM KSDS together with its alternate indexes and paths. There is a base cluster component KSDS (MY.VSAM.DSET), alternate index (MY.VSAM.DSET.AIX), and path (MY.VSAM.DSET.PATH). All these components together are called a VSAM sphere.

Like the cluster components of the KSDS and alternate index, the path does not exist on disk; it is only a catalog entry that relates the base cluster and alternate index together.

All these objects can have any name specified when creating the base component, alternate index, or path.



VRRDS

The previous section introduced VSAM RRDS data sets, and how they must have fixed-length records.

VSAM has another data set: Variable-length Relative Record Data Set (VRRDS). Like an RRDS, the VRRDS can be accessed sequentially or directly using the RRN. However, they can hold variable length records.

Unlike RRDSs, VRRDSs do not have slots. They are in effect VSAM KSDS data sets that are accessed using an RRN instead of a key. Like KSDS data sets, they have both a data component and an index component.