Chapter 2

DATA MANAGEMENT

2.1 FILE ORGANIZATION

The information processed by a computer generally consists of large quantities of data which must be read, written and updated by an application.

Data management is concerned with the possibilities and methods of organizing data in such a way that they are accessible for different applications.

2.1.1 Files

Related data are grouped into files. To enable an application to retrieve data from the file, the information must be stored in the file according to a number of rules, defining the sequence of the data and the way of identifying them. These rules determine the file organization.

A data file need not be a contiguous area on the disk. Separate parts of the data file may reside on one or several volumes.

File Section

A file section is a continuation of the file on a different volume. File sections may reside on different disk types. Up to four file sections are allowed for a file.

File Extent

A file extent is a continuation of the file in a separate physical area on the same volume. Up to 64 extents of one file are allowed per volume. The logical sector number of the first sector of a file extent is always a multiple of 3, and the file extent length is also a multiple of 3 logical sectors and of the block length.

2.1.2 Data-records

Data items holding information on the same subject (e.g. an account-holder, or an article) are grouped into records. A file contains records of the same type. Account-holders records will reside in an account-holder file, article records constitute an article file.
In a PTS system records on a file must all have the same length.

Status Byte

Data management adds a status byte to every record. This byte indicates if a record is "used" or "free".
When a file has been created and formatted, the file is preset with empty records with a status "free". New data records written to the file overwrite these free records and the status byte is set to "used".
When a record is deleted by the application its status is set to "free".
The status byte is not included in the record length for I/O, but it must be taken into account when calculating the blocking factor.

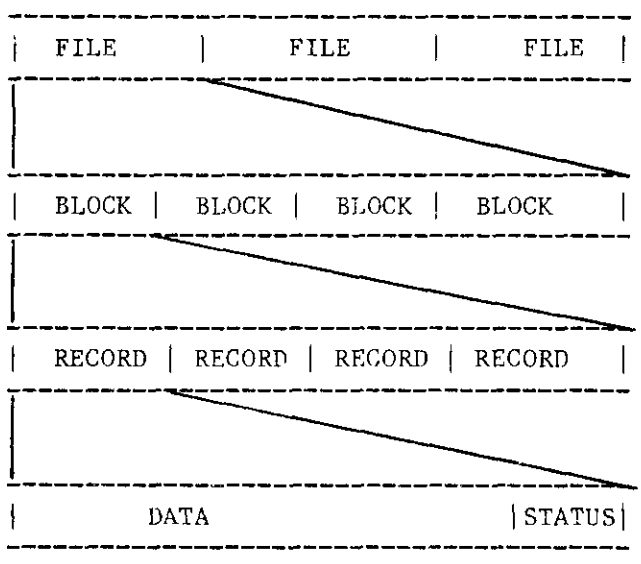

Fig. 2-1 Data Organisation

2.2 RECORD-IDENTIFICATION

Data records on a file are identified by keys. There are two methods:

- Relative Key

  The records of a file are identified by their sequence number in the file. This is the position of the record relative to the beginning of the file, and is called the relative record number or relative key. The first record in the file has relative key 1.
  Every record can always be located by its relative key.

- Symbolic Key

  For the user it may be easier to identify the record by one or more of the data items on it. These are the symbolic keys or record keys. For example, the key of an account holder's record could be the surname, the account number or the user number, and the key of an article record could be the article name or code, or the name of the supplier.

2.2.1 Index File

When the application provides a symbolic key to identify the data record that must be accessed, this has to be converted to a relative key for the system. A table is built with one entry for every record in the data file, holding the symbolic key and the relative key of the data record. This table forms an index to the data file and is called the index file.

Index Levels

In the index file there is one index entry for every data record. For a large data file, this means a long search of the index file before the reference is found.

To reduce search time for a record, the index file may be divided into parts and for every part another index entry may be created. These index entries indicate the range of symbolic keys contained by each part of the index file. Together they constitute the "master index" or the level 1 index. When a record must be located via a symbolic key, the master index is searched first. The found entry points to the part of the index file to be searched and here the pointer to the data record will be found.

2.2.2 Prime Key

At least one of the keys must be unique for each data record. This is the prime key. The index containing the prime key must be defined as the first index when the file is created or opened.
The other keys are called alternate keys, and these need not be unique for one data record.

2.2.3 Duplicate Keys

For the alternate keys, duplicates are allowed: the key may have the same value in several records. For indexed accesses on these records, the first one is found by an indexed direct access and the others are then accessed by indexed sequential operations.
The Return Status "Duplicate Key" will inform the application that the next record has an identical symbolic key.

2.2.4 Curreny

For every task that opens a file, data management keeps a pointer to the current record for the task. The current record is the record last read. The currency is updated by read instructions and it is not affected by write instructions.

The currency allows the application to:
- read the next record
- rewrite the current record
- discard the current record

The currency of the data file is called the Current Record Number or CRN.

If the file is indexed, data management also keeps an index currency. This points to the current index entry: the index entry used for the last read instruction via this index.

2.3 RECORD ACCESS

2.3.1 Access Method

The access method is the way to find a record in the file. If the records are identified in more than one way, there exist several access methods for the same file.

Non Indexed Access

Access on a data file without indexes may be:

- Sequential
  Records are processed in sequence of the relative key. The next record is read or written. For Read Sequential instructions this is the record following the record last read. For Write Sequential instructions this is the first free record in the file, according to the file type. File types are explained in later chapters.

- Direct
  The record to be accessed is indicated by the relative key specified by the program. Records may be accessed directly in any (random) sequence.

- Current
  For Rewrite and Discard instructions the current record may be specified.

Indexed Access

Access on an indexed data file may be:

- Indexed sequential
  The records are read in the sequence in which they appear in the index file, that is, in sequence of the symbolic record key for that index.

- Indexed direct
  The record is identified by a symbolic key, either the prime key or an alternate key, specified by the program. For some instructions this has to be the prime key (see the instruction descriptions in Chapter 6).

  Indexed direct read with a symbolic key specified for which duplicates exist in the file, will access the first record with that key occurring in the file, indicating "Duplicate Key" in the Return Status. The other records with this key may then be read with Read Indexed Sequential.

2.3.2 Examples

Some examples of the different access methods for the file structure in fig. 2-2:

- Sequential Access
  Sequential access on the data file will access the records 'Clayton', 'Shaw', 'Wilcocks', and so on, in the order in which they appear in the data file.

- Indexed Sequential Access
  To access the record in nummeric order of the customer number, which is the prime key, they may be accessed via the index. The records will than be read in the following order: Phyllis Wathke 022; Deborah Williams 043; Francis Dewidt 122; Ronald Williams 207; and so on.

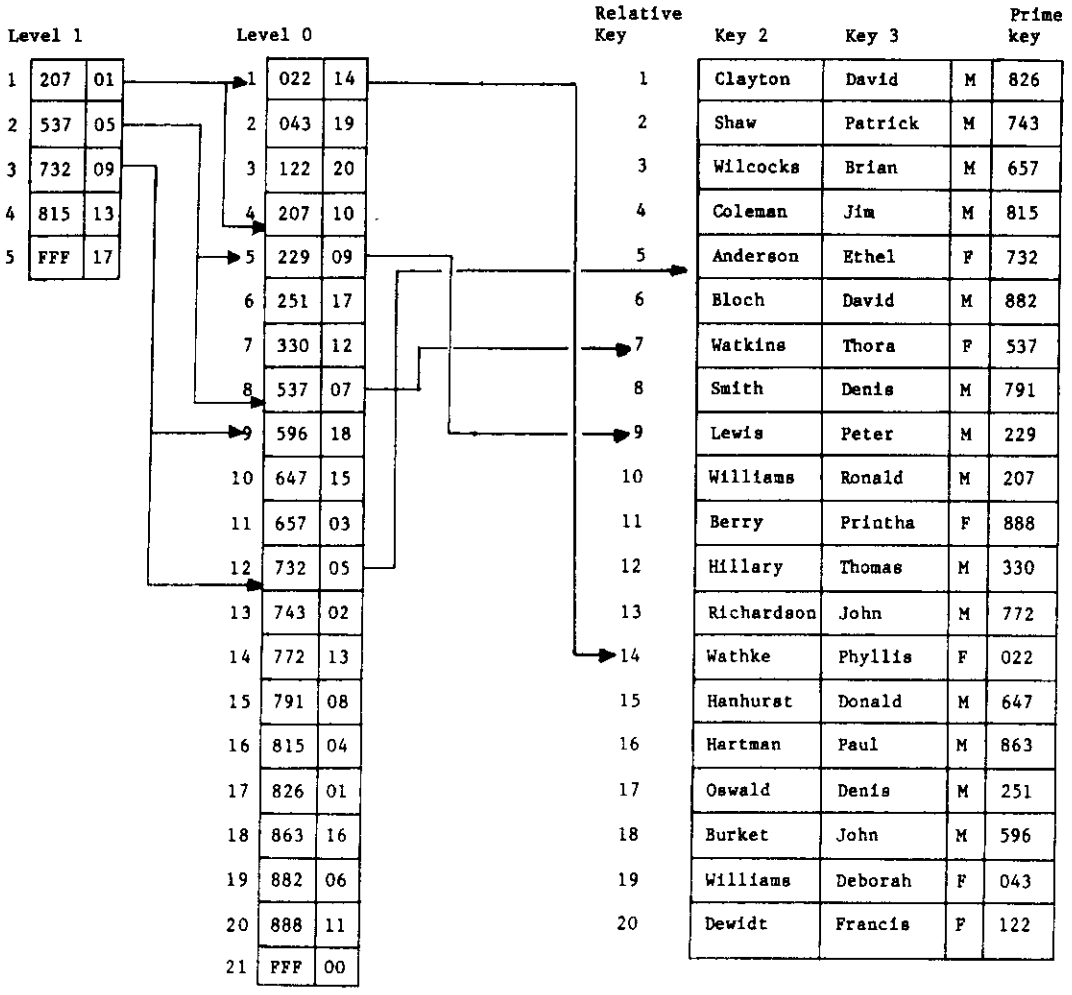

Fig. 2-2 Indexed File

2.4 BLOCKING

Blocking is grouping records into larger units that are transferred during one disk access, because transferring one record per disk access is in most cases not efficient. Transfer always starts on a logical sector boundary, so if the records are shorter or a little longer than 255 bytes, large areas remain unused.

For every transfer the read-write head is positioned at the required sector and transferring the records one by one means that a search time is needed for every record. This takes more time than is necessary, especially when the records are processed sequentially. Better use of time and disk space is made by "blocking" the records. A block is a number of records transferred during one disk access.

2.4.1 Blocking Factor

The number of records per block is the blocking factor. The blocking factor is chosen by the user when the file is created. To choose a blocking factor by which the most efficient use is made of the available disk space, it must be noted that:

- The system adds one status byte to every data record (except for L and X files, see chapter 7). When calculating the blocking factor, 1 must be added to the record length.
- Logical sector length is 256 bytes.
- Blocks always start on a sector boundary, but they may have a block length of several logical sectors.
- File extents always start on a sector with a logical sector number which is a multiple of three.
- The most effective disk access time is obtained when 3 logical sectors are read or written in one access, especially when 16+80 Mb disks are used in the PTS6000 system.
- For large blocks, large block buffers are needed in memory.

2.4.2 Examples

When the record length is 40 bytes a blocking factor of 6 uses 6x(40+1)=246 out of 256 bytes per logical sector.

When the record length is 150 bytes a blocking factor of 3 uses 3x(150+1)=453 bytes out of 512, and every block occupies 2 logical sectors. A blocking factor 5 uses 5x(150+1)=755 out of 768 bytes and every block occupies three logical sectors. The most efficient blocking factor in this case is 5.

2.5 DATA MANAGEMENT FUNCTIONS

2.5.1 File Handling Functions

Data management supports the following functions:

Create File

A new file may be created during runtime. The file must be opened for "Output only" and the application must supply the necessary information such as file name, volume where the file must reside, file size, record length, blocking factor, and the definition of symbolic keys if it is an indexed file. In that case the index filesare also created. As much space as is requested is reserved on the disk and formatted with "free" records.

The following functions can be executed for an existing file:

Open File

An Open file instruction is necessary to initiate a file for access by a task.

Disk files are held on a volume which may contain several different files. Also there may be more than one disk volume on-line at the time, and the data file may have index files to it that reside on a different volume. Therefore, an Open instruction must be executed to tell the Monitor which file is to be opened, on which volume(s) the file exist, how many indexes are to be used and on which volume the index files are found, before the records of a file can be used by the application.

Data Management checks if the task is allowed to open the file, and if there is space in memory for block- and record buffers, currency and protected-record administration.
If all requirements are met the file is opened for the task.

Read Record

Read Record is the instruction to read data from the file. The records are read into the application record buffer.

Rewrite Record

If in the course of a transaction some of the data in a record must be changed, this is done by the application in the application record buffer. The updated record is then rewritten to the file, where it overwrites the old one.

Discard Record

A record which is no longer needed can be discarded. The data is not physically removed from the disk but the status of the record is changed to "free". A free record can not be read by the application.

Write Record

A new data record may be written to the file from the application record buffer. A new record can only be written to a free record in the file.

Extend File

If a number of new records must be written to an existing standard file, the file may be opened for Extend. New records are written to the free part at the end of the file.

Close File

When the task no longer requires access to the file, it must close the file. The space reserved for buffers and administration within the Monitor becomes available for other tasks or other files to be opened. It is especially important to close files which were opened for exclusive access by the task as soon as possible, so that a second task may then open the file.

Delete File

A file no longer needed may be deleted from the disk. The VTOC record for the file will then get the status "free" and the file can no longer be accessed. Only a file that has been opened for exclusive access by a task can be deleted by that task.

2.5.2 Sharability

A number of tasks may be using records of the same files at one time. There must be a protection against simultaneous updating of records by different tasks. Protection is possible on record level and on file level.

Record Protection

- Unprotected
  There is no restriction on concurrent use of the same records by other tasks. "Unprotected" is only allowed when the file has been opened for input only (the records can only be read, not updated or discarded).

- Protected
  When a file is opened protected, a task will hold the records it accesses under Protected Access. No other task can access the record. Other tasks may still access other records on the same file.

  The records are released when the task issues a transaction control instruction (see section 2.7) or closes the file, or when data management releases the records automatically to prevent a deadlock situation.

File Protection

- Exclusive
  Protection on file level means that the file is attached to the task, and no other task can access records of this file. A task can obtain exclusive access to a file by specifying sharability Exclusive when the file is opened. Exclusive access to a file is released by a Close instruction.

  Sharability Exclusive must be specified when the file is created or extended, and when it is to be deleted.

2.5.3 Data Set Declaration

Data management files to be accessed by an application must be defined by a DSET declaration in the data division, in the same way as other I/O devices. This is described in the CREDIT Programmer's Guide for Elementary CREDIT, module M21A.

The DSET declaration links the data set identifier used by the application to the TOSS file code specified during Monitor generation. Data Management file codes must be defined in Special Device Classes.

It is not possible to have common files in CREDIT applications.

2.6 FILE ENLARGEMENT

File enlargement is the addition of another file extent. Both non-indexed and indexed (not in ADM) files can be automatically enlarged during runtime. Automatic enlargement takes place when during Write instructions the last record of the file is written. For the details of automatic enlargement, which are different for each data management package, refer to chapters 7, 8 and 9.

2.6.1 Growth Factor

The size of the added file extent in the case of automatic enlargement is determined by the Growth Factor in the File Descriptor Block. The Growth Factor is set by the user when the file is opened. It represents a percentage of the size of the file when it is opened. From this, the number of records by which the file must be extended is calculated by data management. This number is then rounded upward to obtain a file extent length which is a multiple of three logical sectors and of the block length.

If after the file has been enlarged the end of file is reached again by Write instructions, the file is enlarged again by the same number of logical sectors, for it is the same percentage of the length of the file when it was opened.

The number of sectors by which the file will be enlarged is changed when the file is closed and opened again. A different Growth Factor may then be specified. However, if the Growth Factor remains the same, the percentage will be taken from the new file size and also result in a different file extent length.

Files are only extended if the Growth Factor specified is not zero.

If the file can not be enlarged, the message End of Medium is returned when the end of the file is reached by Write instructions. The file can not be enlarged if:

- The Growth Factor is zero
- There is no free VTOC record available for the new file extent
- The maximum number of file extents (64) on a volume has been reached and no next volume is available
- The maximum number of file extents and file sections has been reached

2.6.2 Example

A file with a size of 200 records is opened. The Growth Factor specified is 10. During Write instructions, the end of the file is reached. A new file extent is automatically created by data management, with a size of 10% of 200 records =20 records.
The sequential write operations are continued until the new end of file is reached. Another file extent is added, with a size of 10% of the original file size (200 records) so again 20 records.

Then the file is closed, and opened again. The size is now 240 records. The Growth Factor specified is still 10. Automatic enlargement will be by 10% of 240 =24 records.

If, when the file is opened for the second time, a Growth Factor of 5 is specified, automatic enlargement will be by 5% of 240 records = 12 records.

Note that this is only an example to explain the mechanism. In reality it would not be advisable to create such small file extents. Also, the size of the additional file extents will be rounded upwards to a multiple of three logical sectors.

2.7 TRANSACTION CONTROL

In a business environment, applications will mostly be designed to execute transactions. A transaction is an elementary business operation. A transaction may include one or more reads, writes and updates of data records in a number of files.

Before and after a transaction the data in the files are in accordance with each other and reflect a real situation. It is said that the files are in a state of integrity. While the transaction is in progress, the files are not in a state of integrity.

2.7.1 Integrity Unit

A series of record accesses on files, at the beginning and end of which the files are in a state of integrity, is called an integrity unit.

One transaction may consist of one or more integrity units.

Example

A data file contains account holders records. A certain amount of money must be withdrawn from the balance of Mr. X and paid into the bank account of Mr. Y.
Before the transaction Mr. X has the money and Mr. Y has not, which is a real situation. Halfway through the transaction, when the money has been withdrawn from the balance of Mr. X and not yet added to the balance of Mr. Y, the data in the file are not in a state of integrity as they do not reflect a real situation.
When the record for Mr. Y. has also been updated the file is again in a state of integrity and the transaction is completed.

The transaction in this example consists of one integrity unit.

2.7.2 Transaction Control Functions

The beginning and the end of a transaction are marked by transaction control functions (Commit). All accesses on the files executed between two transaction control functions belong to one transaction.

Several records of several files may have to be updated during one logical transaction. Before the transaction and after it, when all updates belonging to one logical transaction have been executed completely or not at all, the files are in a consistent state.

Points in the application program where files are in a consistent state are defined by the transaction control functions Commit and Rollback.

COMMIT terminates a transaction or subtransaction. The updates performed are "committed" to the data file, this means that it is no longer possible to undo the transaction. The records involved are released and may now be used by other tasks.

See also the detailed descriptions of transaction control functions for the different packages.