by data. For example, colour codes 1,2, and 3 for black, red and green, or date codes such as 841011 for 11 October 1984. Coding systems must remain internal to the system, however, and programs should convert the codes back into an easily comprehensible form once a field is displayed or entered.

There are two further considerations to bear in mind when determining record lengths. Most systems place a limit on the maximum length available. This can vary from 128 bytes to as much as 2,048 bytes. Additionally, it is often more efficient to choose a length that is a multiple or factor of the sector size — figures such as 64, 128, 256 or 512 are commonly used. This will prevent individual records from being split over more than one sector and therefore reduces the number of disk calls that need to be made.

Random files are generally much easier to handle than serial files. In both systems, you need to keep an up-to-date count of the number of records in a file and quite often the first record in a random access file (often record number 0) is used to store this information and other relevant information such as the file creation date. The rigid field and record structure would be discarded for this record.

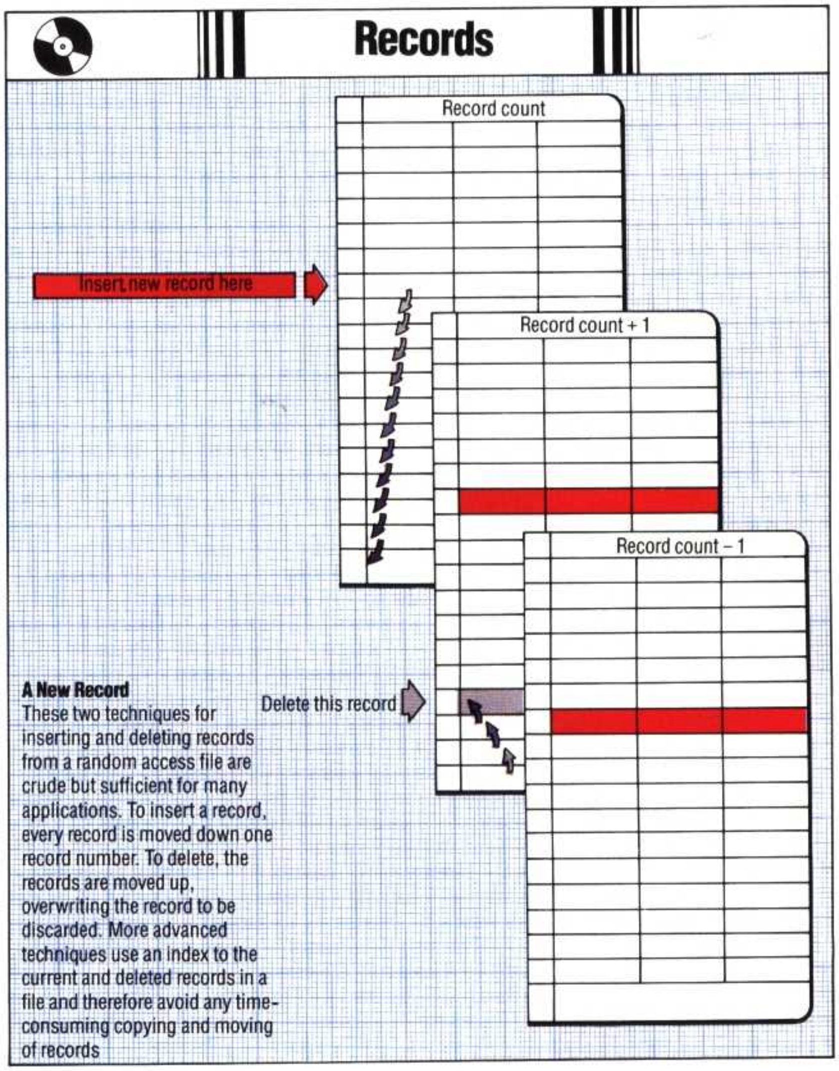
A record can be amended by reading it in, changing it and then writing it back to its location. The record is retrieved by number. Obviously, it is unreasonable to ask a user to remember which record is which by number. So a whole variety of techniques exist for searching and locating particular records, similar in concept to techniques used to search BASIC arrays. Often, one particular field, perhaps a name field, is used as a key to the file. The computer reads in the key field and builds up an index that identifies where various names are stored.

Unindexed random files are often searched record by record just like sequential files. However, if the records are sorted on the key field, fast search methods can be used. Suppose, for example, we wanted to look up 'Jones' in a file sorted by name. We begin by fetching the middle record and discover that the name is 'Phillips'. 'Jones' is before this alphabetically so we can rule out everything after this record. Our next guess is then a record halfway through the first half of the file. The name might be 'Hearst', in which case we need to go forward again and so on. Such techniques can be very sophisticated, and many programs improve performance by keeping large numbers of the most frequently used records in RAM so that they are quickly available. As a result, records can be located and stored within very large files at a high speed.

Deleting and inserting new records can be comparatively slow. The crudest method to delete a record is to copy the record immediately following it into its space, thus overwriting the information in it. Every subsequent record is then copied up one position and finally the record count is reduced by one. In a similar way, a new record can be inserted at any point by moving the last record one number further on and then copying all the records before it and after the number of the new record one space further down. This creates a one-record gap where the new record can be written.

Neither of these techniques is fast, although they are more efficient than similar operations with sequential files. However, insertions and deletions can be made far more quickly if the file has a separate index. When a record is deleted, it is marked as such in the index. The data itself is left unchanged. As new records are added they can be slotted into unused or deleted records and the index updated.

There are two final advantages to consider in the random file system. Firstly, while it is certainly quicker to read and write groups of records together, files can get out of order. Most programs therefore offer a tidy-up facility that sorts records into a logical order and discards deleted records. Secondly, the system of merely marking deleted records as deleted provides a useful safety net, as it is easy to retrieve these records if required. This safety net will operate up to the point when deleted records are overwritten or discarded by a tidy-up program.



A New Record
These two techniques for inserting and deleting records from a random access file are crude but sufficient for many applications. To insert a record, every record is moved down one record number. To delete, the records are moved up, overwriting the record to be discarded. More advanced techniques use an index to the current and deleted records in a file and therefore avoid any time-consuming copying and moving of records

KEVIN JONES