employee records using surnames as the sort key. The hashing algorithm that we will use is: take the ASCII codes of the first four letters and treat them as an eight-digit number, square that number, then take the last four digits of the number as the hash. JONES, therefore, hashes into record 1161, whereas JONQUIL hashes into 0161.

Hashing is very different from an indexed system. With hashing, you can only have one key field (and one hashing algorithm) per file and this is used when first placing the records in the file. Any number of indices can be associated with a particular file and these can be created at any time after or during the file's creation.

Hashing is less flexible than indexing but it is much quicker. To find a particular record, the program just takes the key, hashes it and retrieves that particular record. The time taken to search an index (and indeed to create it in the first place) is therefore dispensed with.

A problem with hashing arises when two records generate the same hash code and therefore should occupy the same position in a file. To avoid this, hashing algorithms are carefully designed so that no two keys (save for identical ones) generate the same hash. Additionally, records are spaced out in the file so that two hashes that are apparently next to each other actually cover a gap of five or so unused records.

We can now clarify our description of a hashing system as follows. When a record is stored, its key is hashed to produce a record number. If that record is occupied, the system looks at the next record sequentially. It can do this for the whole block of five (or whatever) records associated with that hash. When a record is to be retrieved its key is hashed and that group of records is then searched sequentially for an exact match. This may seem to nullify the speed advantage, but what hashing effectively does is to reduce the number of records to look through from perhaps three thousand to five or six.

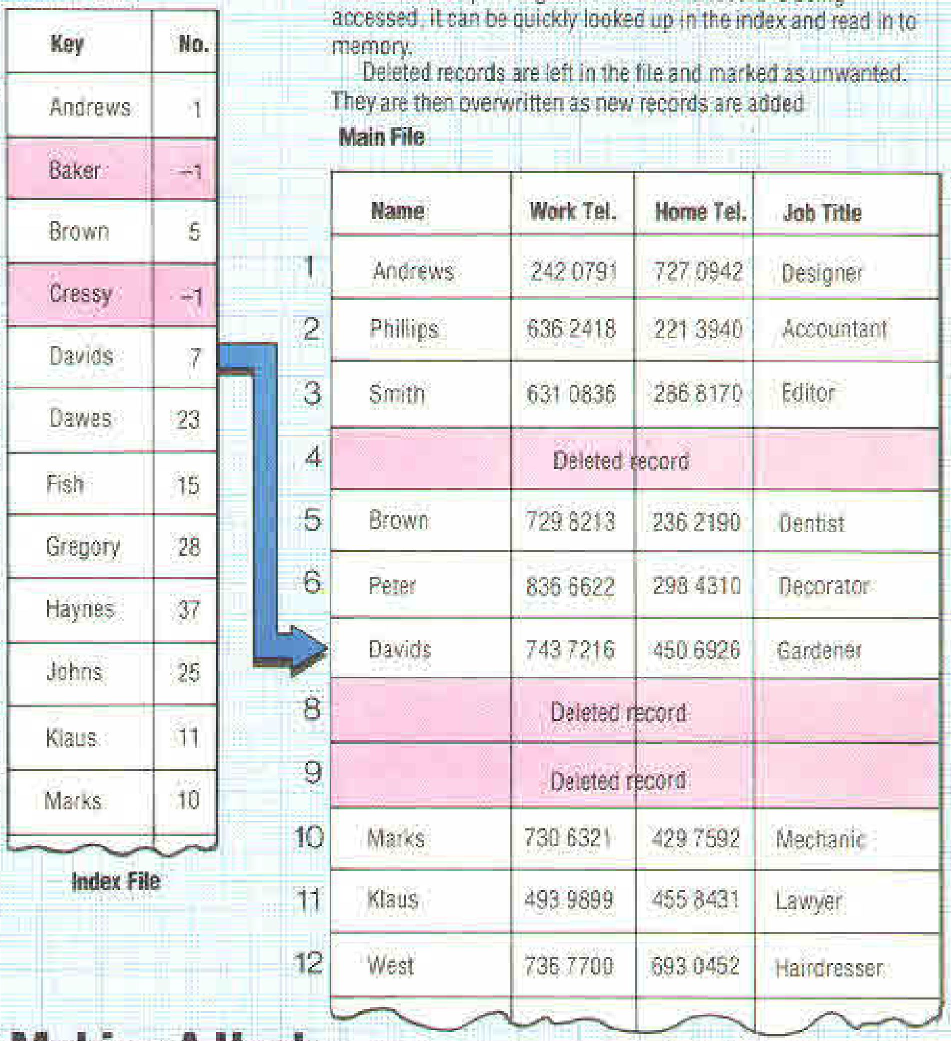
What happens if all five or so records for a particular hash become filled? There are several ways to cope with this, the obvious one being to report a 'file full' message. More often, records that can't be fitted in position in the file are written to a separate overflow file with its own index and incorporated into the main file when possible. Most systems make a determined effort to avoid overflow by habitually keeping hashed files only 80 per cent or less full. This highlights another limitation of hashed access to random files. A hashed file tends to consume more space than if the system used an index.

Hashing also speeds up the deletion of unwanted records. You simply hash the key of the record, do a quick search to locate it exactly and mark its position as unfilled. It will then be overwritten the next time a record with an identical hash is added to the file.

In the final instalment of this series we will look at the BASIC commands necessary to create and access cassette files.

Index Linked

Find 'Davids'



Key	No.
Andrews	1
Baker	−1
Brown	5
Cressy	−1
Davids	7
Dawes	23
Fish	15
Gregory	28
Haynes	37
Johns	25
Klaus	11
Marks	10

Index File

The most common way to access a random file is with an index. This is a list in RAM showing the values for a particular key field with the corresponding records. When a record is being accessed, it can be quickly looked up in the index and read in to memory.

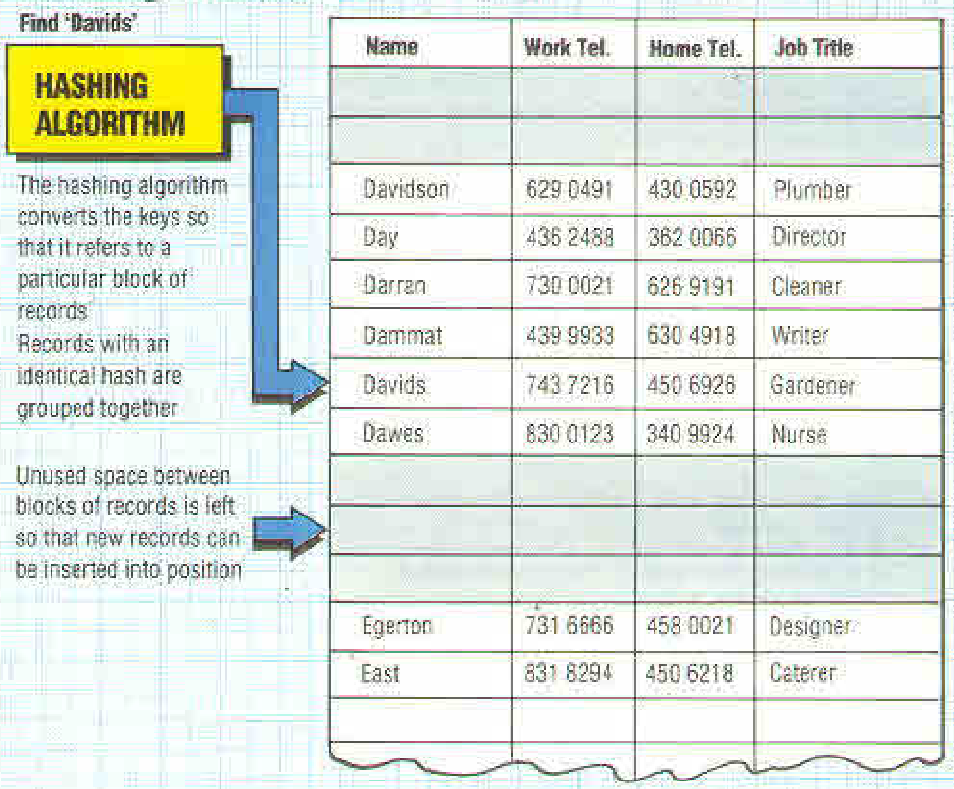
Deleted records are left in the file and marked as unwanted. They are then overwritten as new records are added

Main File

	Name	Work Tel.	Home Tel.	Job Title
1	Andrews	242 0791	727 0942	Designer
2	Phillips	636 2418	221 3940	Accountant
3	Smith	631 0836	286 8170	Editor
4	Deleted record			
5	Brown	729 8213	236 2190	Dentist
6	Peter	836 6622	298 4310	Decorator
	Davids	743 7216	450 6926	Gardener
8	Deleted record			
9	Deleted record			
10	Marks	730 6321	429 7592	Mechanic
11	Klaus	493 9899	455 8431	Lawyer
12	West	736 7700	693 0452	Hairdresser

Making A Hash

Find 'Davids'



The hashing algorithm converts the keys so that it refers to a particular block of records.
Records with an identical hash are grouped together

Unused space between blocks of records is left so that new records can be inserted into position

Name	Work Tel.	Home Tel.	Job Title
Davidson	629 0491	430 0592	Plumber
Day	436 2488	362 0066	Director
Darran	730 0021	626 9191	Cleaner
Dammat	439 9933	630 4918	Writer
Davids	743 7216	450 6926	Gardener
Dawes	830 0123	340 9924	Nurse
Egerton	731 6666	458 0021	Designer
East	831 8294	450 6218	Caterer

Hashed files offer high-speed access to particular records in large random files. However, the system is restrictive and needs careful programming.

The record key is processed into a position in the file with a predetermined hashing algorithm. Each possible hash usually refers to a block of records that can be searched sequentially to find the required record

KEVIN JONES