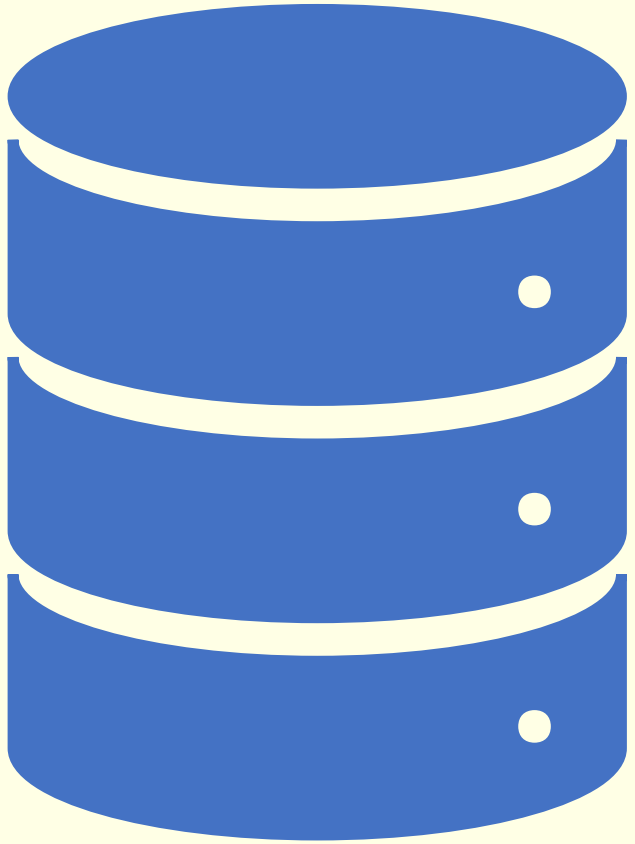
Files, Records, Length and Hashing

- Explain the concept of a relational database
- Define the terms: flat file, entity, attribute, primary key, foreign key, secondary key, entity relationship modelling, referential integrity



Recap

Key term

Record A single unit of information in a database. It is normally made up of *fields*. So a student file would be made up of many records. Each record is about one student and holds fields such as student number, surname, date of birth, gender, and so on.

Flat File Databases

A flat file database consists of data stored within a plain text file.

Records are stored one per line and each attribute of the record is separated by a delimiter – often in the form of a comma or tab. Often saved as a CSV.

A flat file is the simplest form of database and simplicity is its biggest advantage.

Typical uses:

- Storing contact details
- Small product database
- Music files

Example

A typical example of a flat-file database is an address book. Here is a view of part of one:

First name	Last name	Telephone	Street	City	Post code	DOB
Claire	Pate	1 55 791 7964-8421	1434 Aenean Road	Iowa City	K3I 1RF	6/28/1999
Virginia	Landry	1 61 306 9087-9418	404 Morbi Road	Rock Island	EI3O 7QR	1/23/1974
Orli	Goodwin	1 51 119 4068-1665	704-6375 Varius St.	Lynwood	CG12 9LQ	9/26/1984
Callie	Hodge	1 70 829 9014-9968	PO Box 362, 5198 Vulputate, St	Wichita Falls	D1Z 9AN	07/05/1978
Rhonda	Pugh	1 44 202 4884-7705	PO Box 250, 7653 Fusce Road	West Covina	S5 9OD	6/23/1984
Dara	Goff	1 70 115 3175-0607	844-4722 Felis St	Knoxville	KE9C 7XR	10/03/1999

You can easily understand the concept of a flat-file database by envisaging it as a spreadsheet or document table.

Serial File

- Data is stored in the order in which it is entered.
- For example, a text file to store a playlist
- This is a database containing only a single table of information
- Each new record is added to the end of the file
- Simple way to store date, if all we want to do is keep a record of songs in no particular order.

main.py playlist.txt

```
1 I Gotta Feeling;The Black Eyed Peas;4:05
2 Hey Brother;Avicii;4:15
3 This is the life;Amy MacDonald;3:06
4 Wolrd, Hold On;Bob Sinclar;6:41
5 Paradise;Coldplay;4:23
6 Memories;David Guetta;3:30
7 Hot 'n Cold;Katy Perry;3:40
8 Our House;Madness;3:12
9 Timber;Pitbull;3:25
```

Python example

```
file = open("file.csv","r")
for line in file:
    print( line )
file.close()
```

```
file = open("file.csv,"r")
for line in file:
    fields = line.split("; ")
    field1 = fields[0]
    field2 = fields[1]
    field3 = fields[2]
    print(field1 + " " + field2 + " " + field3)
```

```
4
5  #Repeat for each song in the text file
6  for line in file:
7
8    #Let's split the line into an array called "fields" using the ";" as a separator:
9    fields = line.split(";")
10
11   #and let's extract the data:
12   songTitle = fields[0]
13   artist = fields[1]
14   duration = fields[2]
15
16   #Print the song
17   print(songTitle + " by " + artist + " Duration: " + duration)
18
19 #It is good practice to close the file at the end to free up resources
20 file.close()
```

main.py | playlist.txt

```
1 I Gotta Feeling;The Black Eyed Peas;4:05
2 Hey Brother;Avicii;4:15
3 This is the life;Amy MacDonald;3:06
4 Wolrd, Hold On;Bob Sinclar;6:41
5 Paradise;Coldplay;4:23
6 Memories;David Guetta;3:30
7 Hot 'n Cold;Katy Perry;3:40
8 Our House;Madness;3:12
9 Timber;Pitbull;3:25
```

Drawbacks of serial files

- To locate a particular record, it is necessary to start at the beginning of the file and examine each record in turn until the required record is found or the end of the file is reached.
- If the file is large then this could take some time!

Sequential File

- This is when the data in a file can be sorted by a field.
- For example, sorting the records by the 'Amount' field

Transactions.csv

```
Sender,Recipient,Amount

Sam,George,150

Billy,Sam,100

George,Billy,50
```

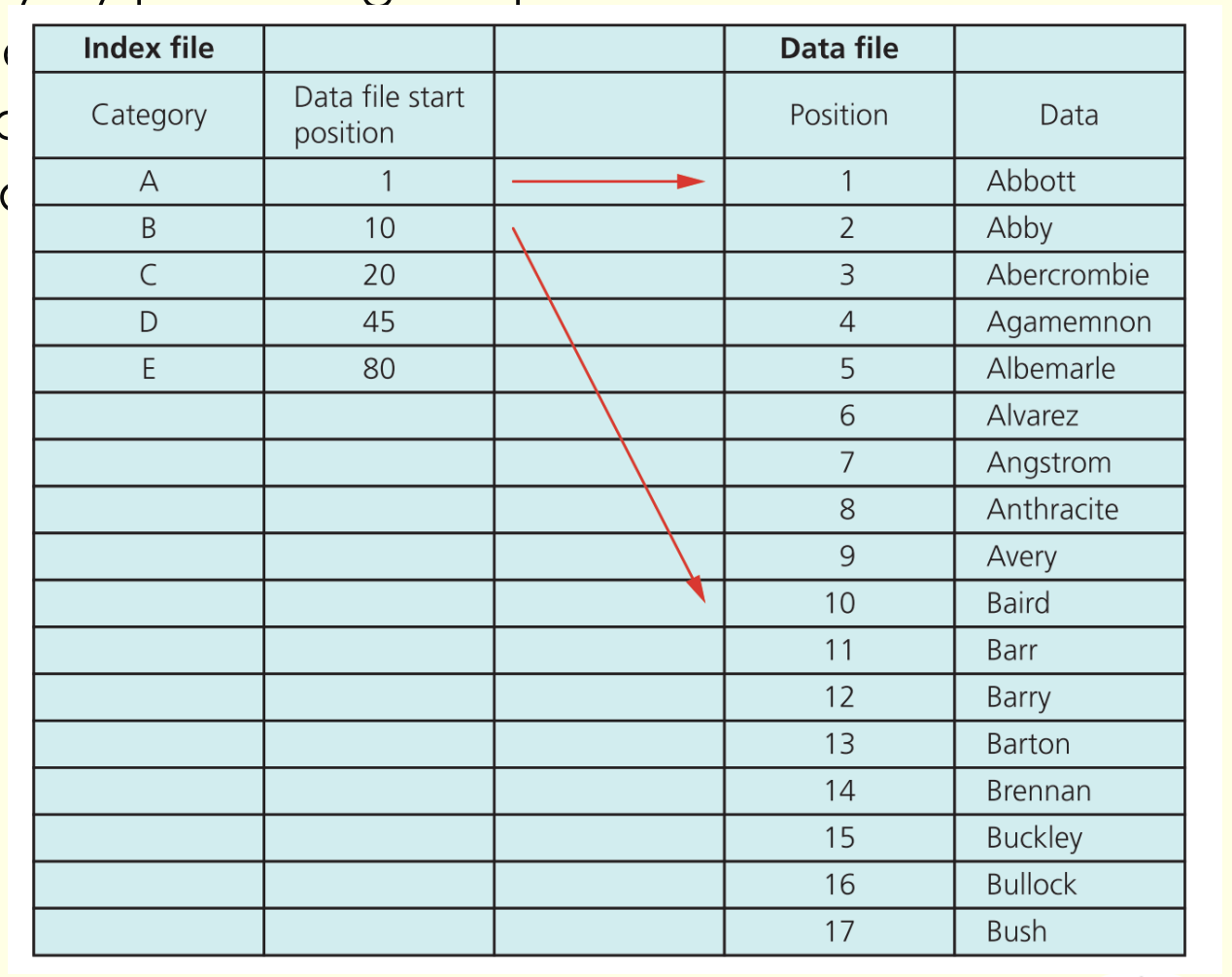
- This makes searching a little easier as the records are ordered, alphabetically or numerical order, for example by ordered by Amount

Drawbacks of sequential files

- In order to generate a sequential file, at intervals the data in the file has to be sorted.
- This would involve writing the date in order to another file.
- The file has to be sorted before searched.

Searching using Indexes

- Sequential files can be searched more quickly by producing a separate index file
- The data is divided up into categories, e.g. n
- Each category is linked to a position in the da
- The number of records that must be searched



Index file			Data file	
Category	Data file start position		Position	Data
A	1		1	Abbott
B	10		2	Abby
C	20		3	Abercrombie
D	45		4	Agamemnon
E	80		5	Albemarle
			6	Alvarez
			7	Angstrom
			8	Anthracite
			9	Avery
			10	Baird
			11	Barr
			12	Barry
			13	Barton
			14	Brennan
			15	Buckley
			16	Bullock
			17	Bush

Drawbacks of Flat file databases

- Duplicate data is stored so storage is wasted ie. Data redundancy
- Changes to data may require rewriting the entire file

Questions & Answers

1. Would an address book laid out like this be useful for:

(a) storing details of your friends

(b) storing customer details for a large online trading organisation?

2. What are the good and bad points of using a flat-file database for these purposes?

Fixed and variable length fields

Fields in records can be fixed or variable in length, and this in turn gives rise to fixed or variable length records.

Fixed – each field has the same number of bytes in length – easy to program but wasteful of space.

Variable length records are more efficient in terms of space (memory), but can be harder to process.

A common file type with variable length records is the CSV file.

CSV stands for comma-separated variable, where each field is separated by a comma. An obvious drawback with this approach is that the data within a field must not contain a comma.

Some systems allow a different separator to be used or for a comma within the data to be flagged as data in some way.

Here is a possible structure of part of a student record in CSV format, showing surname, forename, gender and student number.

S	m	i	t	h	,	J	o	h	n	,	m	,	2	1	6	6	8	,

Hashing

- Hashing is a method of transforming a string of characters in a record into a shortened form that can be used as a disk address.
- This shortened form (has value) can be used to access a record from a database more quickly than by using the complete string.
- Typically, multiple records can usually produce some hash values that are the same.
- In this case, the data is located in the next available space (or block) on the storage medium, so some serial searching may be necessary.

Hashing Example

- Account number 2563546 generates the disk address 546. This leads to a block of records beginning at position 546. The disk address 546 is accessed and the record is written at that location.
- Of course, the account number 5756546 will also generate the same address. In this case, if the position is already occupied, the record is written to the next sequentially available location.
- If the block is full, then any records that generate that address will be written to an overflow area specially designated for such data collisions.

Question

Write an **algorithm** that accepts a seven-digit account number then finds an appropriate three-digit disk storage location. Make sure that you make provision for the storage block being full.