Problem 1: Fixed-length and variable-length records

1. 3783958 | 1297015 | BEST-ACTRESS# | 2017

2. '3783958' - 7 one-byte characters
+ '1297015' - 7 one-byte characters
+ 'BEST-ACTRESS' - 23 one-byte characters
+ 2017 - 1 two-byte integer
7 + 7 + 23 + (1 x 2) = 28
Answer = 39 bytes

3. 7 | 3783958 | 7 | 1297015 | 12 | BEST-ACTRESS | 2 | 2017

4. 7 - 1 two-byte integer field
+ '3783958' - 7 one-byte characters
+ 7 - 1 two-byte integer field
+ '1297015' - 7 one-byte characters
+ 12 - 1 two-byte integer field
+ 'BEST-ACTRESS' - 12 one-byte characters
+ 2 - 1 two-byte integer field
+ 2017 - 1 two-byte integer field
(1 x 2) + 7 + (1 x 2) + 7 + (1 x 2) + 12 + (1 x 2) + (1 x 2) = 36
Answer = 36 bytes

5. 8 | 15 | 22 | 34 | 3783958 | 1297015 | BEST-ACTRESS | 2017

6. 8 - 1 two-byte integer field
+ 15 - 1 two-byte integer field
+ 22 - 1 two-byte integer field
+ 34 - 1 two-byte integer field
+ '3783958' - 7 one-byte characters
+ '1297015' - 7 one-byte characters
+ 'BEST-ACTRESS' - 12 one-byte characters
+ 2017 - 1 two-byte integer field
(1 x 2) + (1 x 2) + (1 x 2) + (1 x 2) + 7 + 7 + 12 + (1 x 2) = 36
Answer = 36 bytes

7. In a fixed-length record format, the following procedure can be used to extract the type value from the Oscar record:
The system finds the value for the type value column by finding the record and proceeding to byte 15 which is the start of the type value field, reading any data that exists within the next 23 bytes, capturing the data, and returning the value of 'BEST-ACTRESS'.

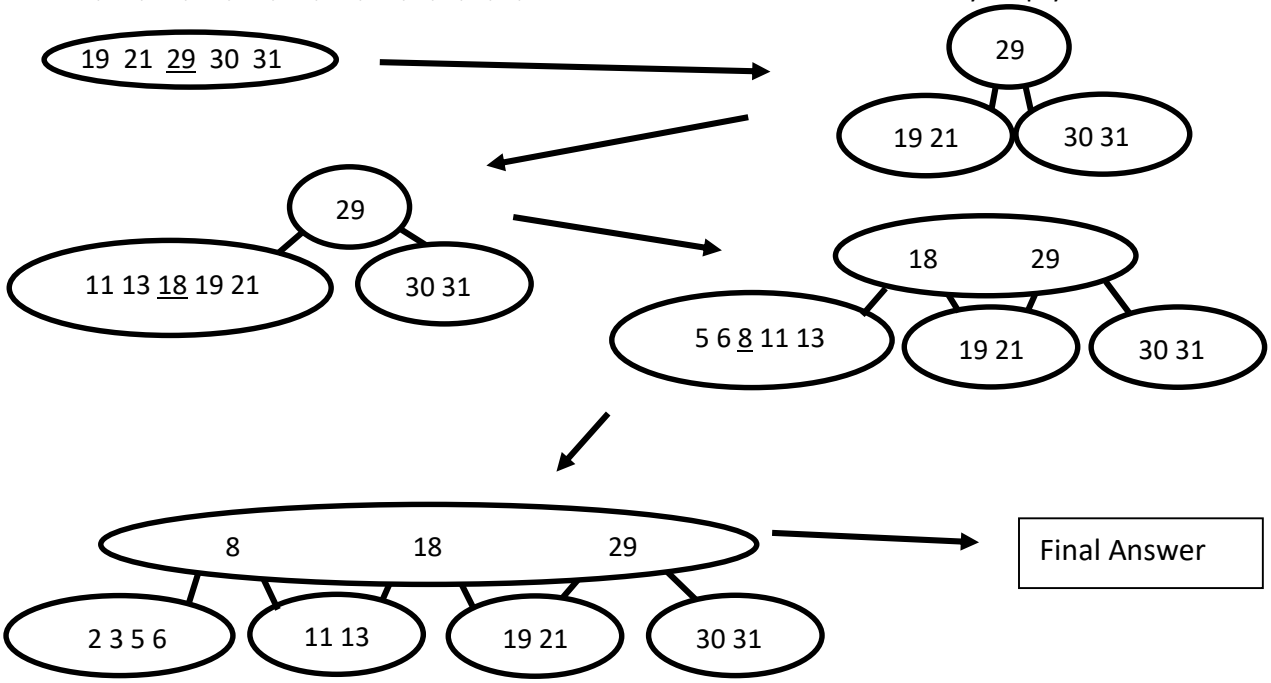
In a variable-length record format in which each field is preceded by its length, the following procedure can be used to extract the type value from the Oscar record:
The system finds the value for the type value column by proceeding to the end of the 2nd field at byte 18, capturing the two bytes of integer data that indicates how large the type field will be (12 bytes), reading in the next 12 bytes of data, and returning the value of 'BEST-ACTRESS'.

In a variable-length record that begins with a header of offsets format, the following procedure can be used to extract the type value from the Oscar record:
The system finds the value for the type value column by reading the data in the third offset (22 bytes), reading the data in the fourth offset (34 bytes), proceeding to the 22nd byte which is the start of the type value field,
capturing the data until it reaches the 34th byte, and returning the value of 'BEST-ACTRESS'.

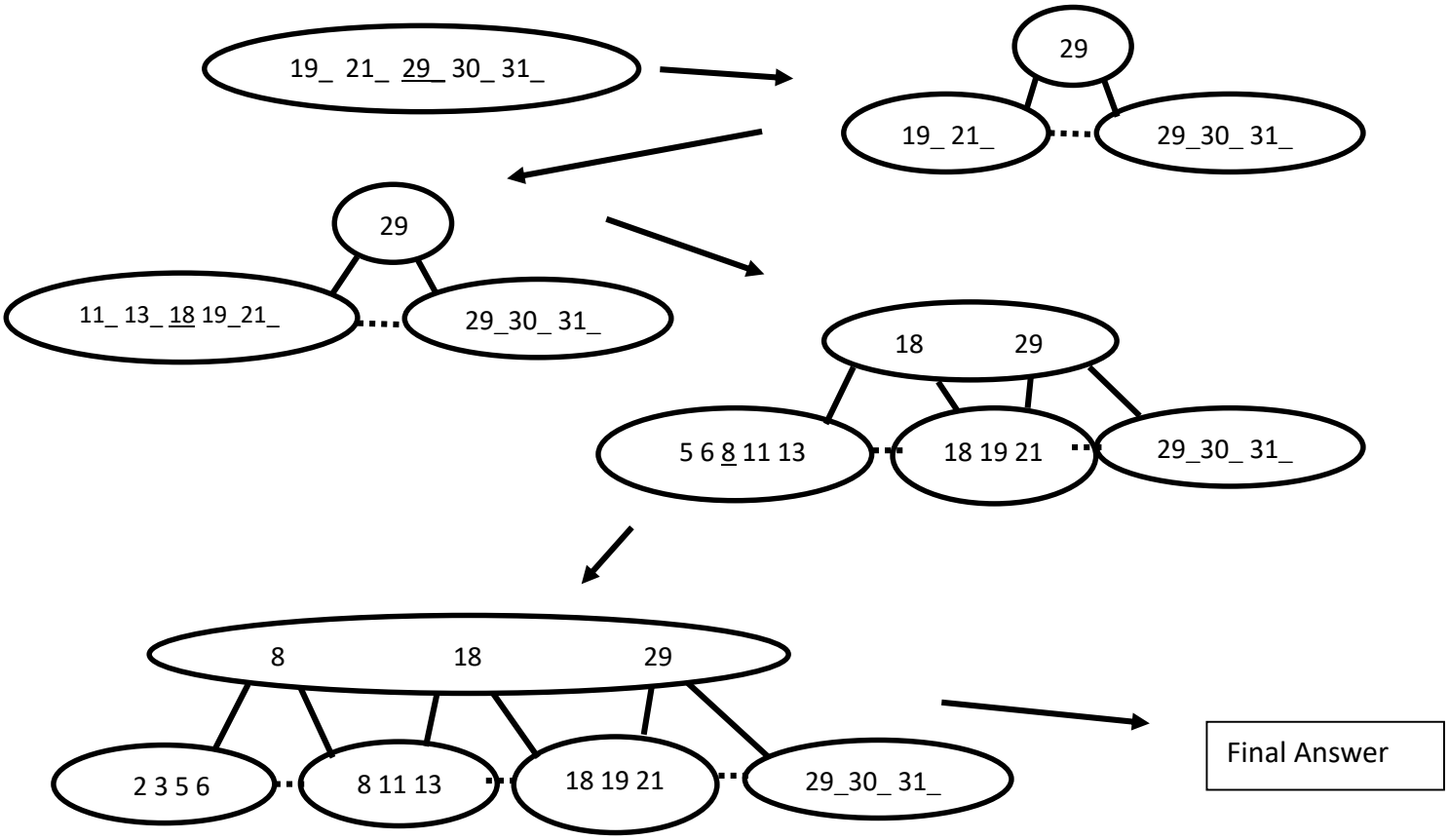
8. One advantage of using a fixed-length record format in the Oscar table is that it simplifies the procedure used to access a given field. The offsets are consistent and the system does not require bookkeeping per record to determine where fields begin and end. A disadvantage of using a fixed-length record format is the wasted storage space allocated to the 23-character VARCHAR attribute which would regularly contain unused storage bytes. Conversely, an advantage of using a variable-length record format in the Oscar table in which each field is preceded by its length is that the system only allocates the space needed for a given record and reduces overall wasted space. One disadvantage of using a variable-length record format in which each field is preceded by its length is that this complicates the access procedure to reach a given record. Similarly, variable-length records that begin with a header of offsets format are advantageous in the sense that they reduce wasted storage space by only allocating the space needed for a given record. A similar disadvantage in variable-length records that begin with a header of offsets is the increased processing time required to access a given field due to the computations that will occur in the header of offsets.

Problem 2: Index structures

1. 31, 30, 29, 21, 19, 18, 13, 11, 8, 6, 5, 3, 2 – Insert these values into an initially empty B-tree order 2



2. 31, 30, 29, 21, 19, 18, 13, 11, 8, 6, 5, 3, 2 – Insert these values into an initially empty B+tree order 2



3. 31, 30, 29, 21, 19, 18, 13, 11, 8, 6, 5, 3, 2 – Insert values into a hash table that uses linear hashing

00 = 0	~~30,~~ ~~18,~~ 8,
01 =1	~~31~~, 29, 21, ~~19~~, 13, 5
10 =2	30, 18, 6, 2
11=3	31, 19, 11, 3

h(31)=31->111<u>1</u>
h(30)=30->111<u>0</u>
h(29)=29->1110<u>1</u>
h(21)=21->1010<u>1</u>
h(19)=19->1001<u>1</u>
h(18)=18->1001<u>0</u>
h(13)=13->110<u>1</u>
h(11)=11->101<u>1</u>
h(8)=8->100<u>0</u>
h(6)=6->11<u>0</u>
h(5)=5->10<u>1</u>
h(3)=3->1<u>1</u>
h(2)=2->1<u>0</u>

n <- # buckets 2

I =[log2n]2

f >2n

00 = 0	8
01 =1	29, 21, 13, 5
10 =2	30, 18, 6, 2
11=3	31, 19, 11, 3

Final Answer