**10.4 Consider the deletion of record 5 from the file of Figure 10.6. Compare the**

**relative merits of the following techniques for implementing the deletion:**

**a. Move record 6 to the space occupied by record 5, and move record**

**7 to the space occupied by record 6.**

**b. Move record 7 to the space occupied by record 5.**

**c. Mark record 5 as deleted, and move no records.**

a. Although moving record 6 to the space for 5, and moving record 7 to the space for 6, is the most straightforward approach, it requires moving the most records, and involves the most accesses.

b. Moving record 7 to the space for 5 moves fewer records, but destroys any ordering in the file.

c. Marking the space for 5 as deleted preserves ordering and moves no records, but requires additional overhead to keep track of all of the free space in the file. This method may lead to too many “holes” in the file, which if not compacted from time to time, will affect performance because of reduced availability of contiguous free records.

**10.5 Show the structure of the file of Figure 10.7 after each of the following**

**steps:**

**a. Insert (24556, Turnamian, Finance, 98000).**

**b. Delete record 2.**

**c. Insert (34556, Thompson, Music, 67000).**

a.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Up 4 | | | | Header |
| 10101 | Srinivansan | Comp.sci | 65000 | Record0 |
| 24556 | Turnamian | Finance | 98000 | Record1 |
| 15151 | Mozert | Music | 40000 | Record2 |
| 22222 | Einstein | Physics | 95000 | Record3 |
|  |  |  | Up6 | Record4 |
| 33456 | Gold | Physics | 87000 | Record5 |
|  |  |  |  | Record6 |
| 58583 | Califieri | History | 62000 | Record7 |
| 76543 | Singh | Finance | 80000 | Record8 |
| 76766 | Crick | Biology | 72000 | Record9 |
| 83821 | Brandt | Comp.sci | 92000 | Record10 |
| 98345 | Kim | Elec.eng | 80000 | Record11 |

b.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Up 2 | | | | header |
| 10101 | Srinivansan | Comp.sci | 65000 | Record0 |
| 24556 | Turnamian | Finance | 98000 | Record1 |
|  |  |  | Up4 | Record2 |
| 22222 | Einstein | Physics | 95000 | Record3 |
|  |  |  | Up6 | Record4 |
| 33456 | Gold | Physics | 87000 | Record5 |
|  |  |  |  | Record6 |
| 58583 | Califieri | History | 62000 | Record7 |
| 76543 | Singh | Finance | 80000 | Record8 |
| 76766 | Crick | Biology | 72000 | Record9 |
| 83821 | Brandt | Comp.sci | 92000 | Record10 |
| 98345 | Kim | Elec.eng | 80000 | Record11 |

c.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Up 4 | | | | header |
| 10101 | Srinivansan | Comp.sci | 65000 | Record0 |
| 24556 | Turnamian | Finance | 98000 | Record1 |
| 34556 | Thompson | Music | 67000 | Record2 |
| 22222 | Einstein | Physics | 95000 | Record3 |
|  |  |  | Up6 | Record4 |
| 33456 | Gold | Physics | 87000 | Record5 |
|  |  |  |  | Record6 |
| 58583 | Califieri | History | 62000 | Record7 |
| 76543 | Singh | Finance | 80000 | Record8 |
| 76766 | Crick | Biology | 72000 | Record9 |
| 83821 | Brandt | Comp.sci | 92000 | Record10 |
| 98345 | Kim | Elec.eng | 80000 | Record11 |

**10.14 In the variable-length record representation, a null bitmap is used to**

**indicate if an attribute has the null value.**

**a. For variable length fields, if the value is null, what would be stored**

**in the offset and length fields?**

**b. In some applications, tuples have a very large number of attributes,**

**most of which are null. Can you modify the record representation**

**such that the only overhead for a null attribute is the single bit in**

**the null bitmap.**

a. It does not matter on what user store in the offset and length fields since we are using a null bitmap to identify null entries. But It would make sense to set the offset and length to zero to avoid having arbitrary values

b. User should be able to locate the null bitmap and the offset and length values of non-null attributes using the null bitmap

**10.19 Give a normalized version of the *Index metadata* relation, and explain why**

**using the normalized version would result in worse performance.**

Give a normalized version of the Index metadata relation, and explain why using the normalized version would result in worse performance. The Index-metadata relation can be normalized as follows Index-metadata (index-name, relation-name, index-type, attrib-set) Attribset-metadata (relation-name, attrib-set,

**10.20 If you have data that should not be lost on disk failure, and the data are**

**write intensive, how would you store the data?**

Use RAID. Using redundancy features,

Make a copy and use it.