TDT4145: Data Modelling and Database Systems

Exercise 4: “B-trees, hashing, physical design, query evaluation”

David Rozas Domingo
Miguel Bono Tur

<u>TASK 1</u>

a)

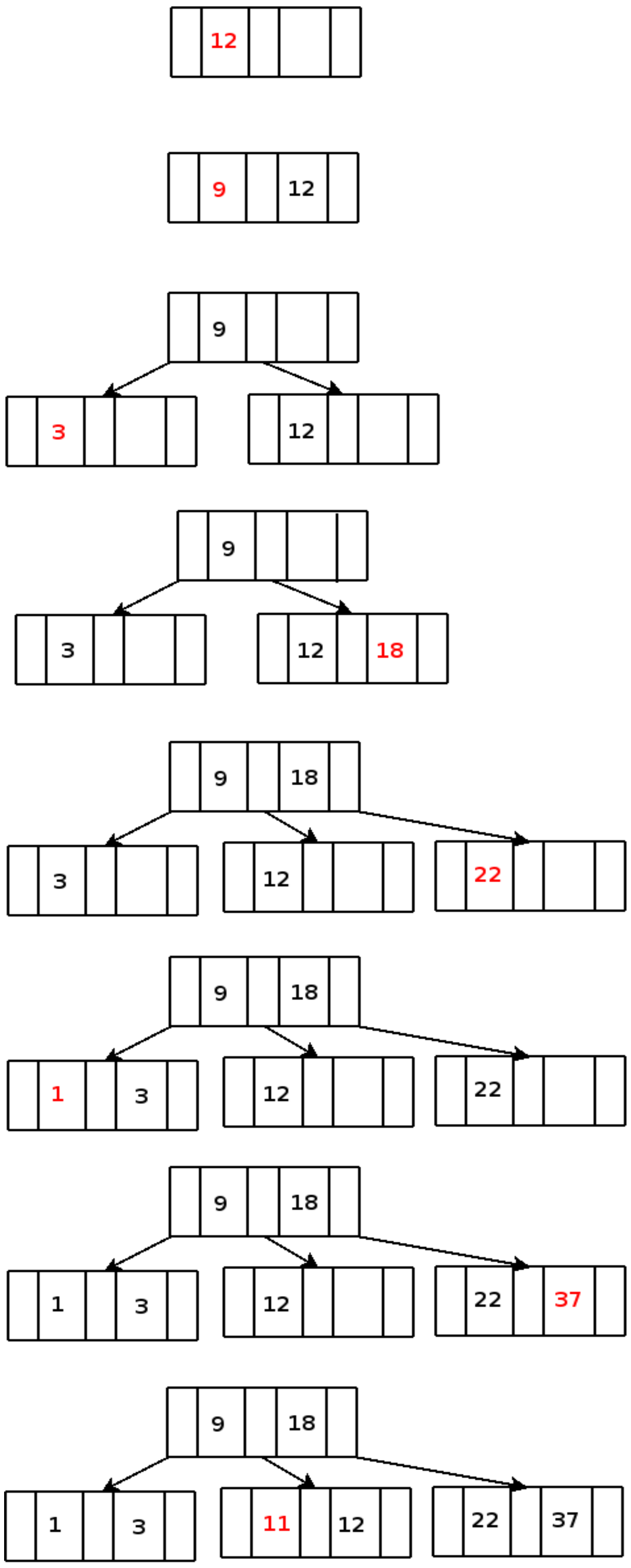
If we suppose F=100 and a tree of height two we would have 10000 entries, so we need at least a tree of height three.

A que se refiere después, a cuántos nodos intermedios?. Sabes cómo calcularlos?.

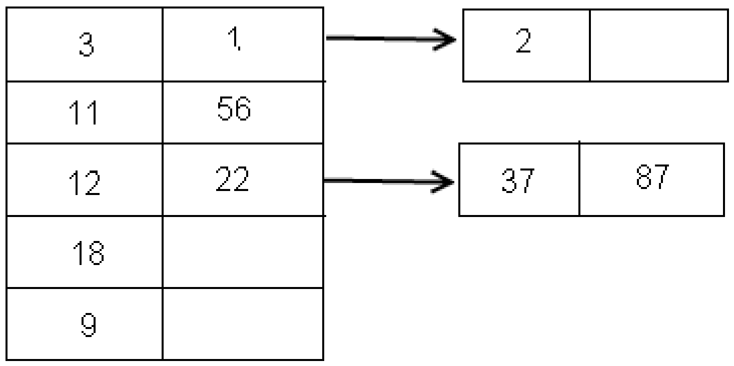
b)
If we suppose that the leaf page has sufficient space for the new record, we have only to locate and modify the page

NUMBERS????
Supongo que en realidad aquí tienes que reestructurar, porque luego te preguntan un update, y si no...es lo mismo!.
c)

d)


12
9 12
9; 3; 12
9; 3; 12 18
9 18; 3; 12; 22
9 18; 1 3; 12; 22
9 18; 1 3; 12; 22 37
9 18; 1 3; 11 12; 22 37

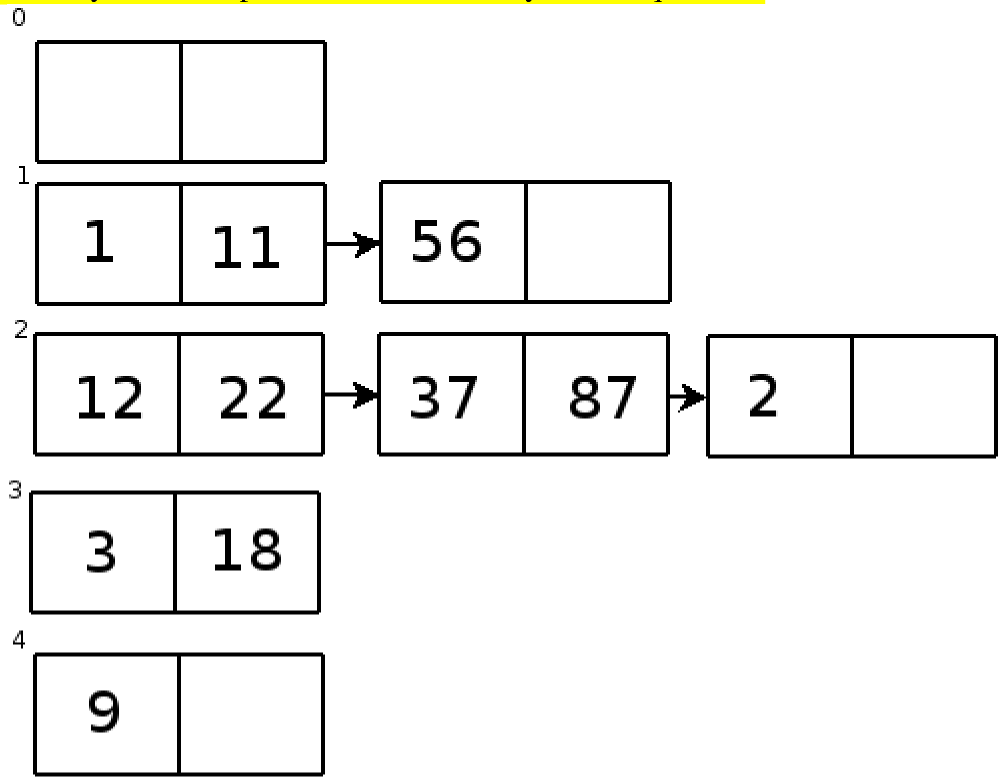
TASK 2



?????

¿Qué has tenido en cuenta para hacer las inserciones?.

Tal y como yo lo veo, aplicando la función key mod 5, quedaría:



TASK 3

A)

Due to the most common operation is a search with equality selection, the database system which support this kind of operations more efficiently is the hash-based indexing, where the search key would be <winename>, or a composite <winename,...>. This is because in this case of systems the cost of identifying the the page is even less than in a tree, but unfortunately this system is inneficient for range selections.

B)

We consider initiallly the creation of:

- An index on ssn
- A composite index <name, depname>

But due to the big amount of inserts operations (40%), we consider that the first case is not going to improve the performance.

About the second, the big amount of this kind of operations shows that a composite index can be very useful, even if it is going to be neccessary to update the index in the insertions, it is worth it.

Pfff, se supone que habrá que hacer cuentas, pero no tenía ni idea...así que he hablado en gral.

TASK 4

Exercise 8.9

The main conclusions about the basic file organizations is that each one have advantages and disadvantages. Depend of the situation, each one of them will be more suitable. The choice of appropriate structures for a given data set can have a significant impact upon performance. An unordered file is best if only full file scans are desired. A hash indexed file is best if the most common operation is an equality selection. A sorted file is best if range selections are desired and the data is static; a clustered B+ tree is best if range selections are important and the data is dynamic. n unclustered B+ tree index is useful for selections over small ranges, especially if we need to cluster on another search key to support some common query.

1. Using these fields as the search key, we would choose a sorted file organization or a clustered B+ tree depending on whether the data is static or not.
2. Heap file would be the best fit in this situation.
3. Using this particular field as the search key, choosing a hash indexed file would be the best.

Es copy&paste descarado de las soluciones, ¿deberíamos cambiar el texto un poco?.

Exercise 8.10

1. If we use the alternative (1), it is clustered by definition, alternative (1) is called a primary index, and primary index includes the primary key.
2. For examples in clustered files, the inserts and deletes are efficient.
3. For example in unclustered tree index the scans and range searches with many matches are slow.
4. Range search are equal in heaps and in unclustered hash index.

Exercise 12.6

INDEX NESTED LOOPS JOIN

Formula:
Cost = Ntuples *(1+1.2)

Let us say that scan R cost 1000. Let us say also that there are 100*1000 tuples in R. For each tuple, retrieving the index page containing the rid of the matching S tuple cost 1.2 I/Os (on average). Moreover, we have to retrieve the sailors page containing the qualifying tuple. Therefore, we have 100000*(1+1.2) I/Os to retrieve matching S tuples. The total cost is 221000 I/Os.

SORT-MERGE JOIN

Formula:
Cost = Sort1+Sort2+Tscan1+Tscan2

Assuming that we can sort R ans S in two passes. Consider the join of the tables R and S. The cost of scanning R are 1000. Because we read and write R in each pass, the sorting cost is 2*2*1000=4000 I/Os. Similarly, we can sort S (cost of scanning 500), the cost is 2*2*500=2000 I/Os. The second phase of the sort-merge join algorithm requires an additional scan of both tables. Thus the total cost is 4000+2000+1000+500=7500 I/os.

.