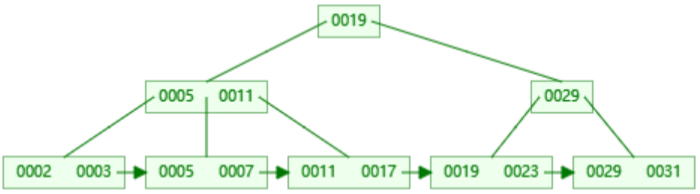
数据库week10

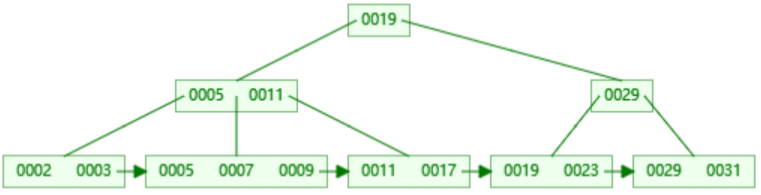
14.3[a]

> Construct a B+-tree for the following set of key values:(2, 3, 5, 7, 11, 17, 19, 23, 29, 31)
>
> Assume that the tree is initially empty and values are added in ascending order.Construct B+-trees for the cases where the number of pointers that will fit in one node is as follows:
>
> Four

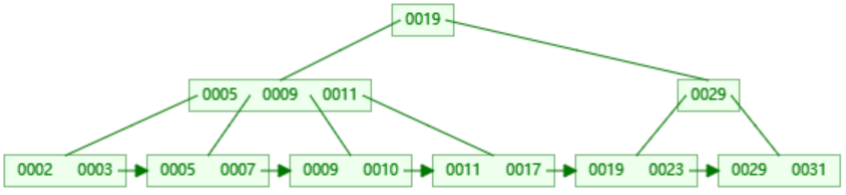


14.4

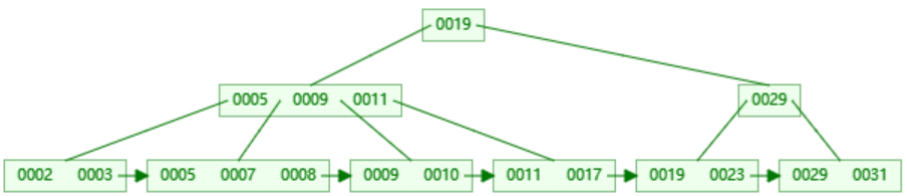
> For each B+-tree of Exercise 14.3, show the form of the tree after each of the following series of operations:
>
> a. Insert 9.



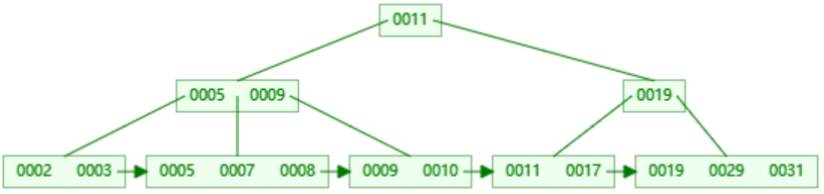
> b. Insert 10.



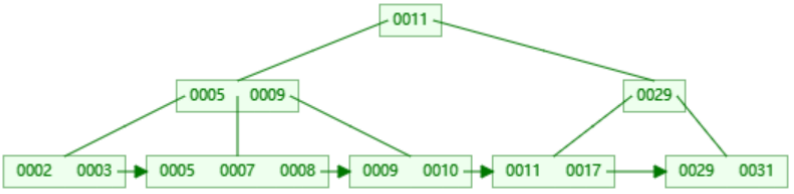
> c. Insert 8.



> d. Delete 23.



> e. Delete 19.



14.11

> In write-optimized trees such as the LSM tree or the stepped-merge index, entries in one level are merged into the next level only when the level is full.Suggest how this policy can be changed to improve read performance during periods when there are many reads but no updates.

If there have been no updates in a while, but there are a lot of index look ups on an index, then entries at one level, say i, can be merged into the next level, even if the level is not full. The benefit is that reads would then not have to look up indices at level i, reducing the cost of reads.

24.10

> The stepped merge variant of the LSM tree allows multiple trees per level. What are the tradeoffs in having more trees per level?.

- Reduces write cost compared to LSM tree.

  The benefit of the stepped-merge index scheme as compared to the basic LSM tree is that index entries are written out only once per level. With the basic LSM tree, each time a tree at level L_i is merged into a tree at level L_{i+1}, the entire contents of the L_{i+1}tree is read and written back to disk.

- But queries are even more expensive since many trees need to be queries.

  More trees per level can initially degrade read performance because the database engine must search through more trees to find the data.Lookups would then have to pay a high price,since they would have to search through each of the tree structures, incurring separate I/O costs on each search.