Ian Boulis
SAT3210
Dec 5, 2021

Assignment 5

1. 14.16
   When is it preferred to use a dense index rather than a sparse index?

The primary difference between a dense and sparse index is that in a dense index, for each search key there exists an index entry. This means that a dense index is preferred when the data is sorted on a non-index field, as a dense index has a record for every data entry. Dense indexes are also preferred when there is not much available memory, as a dense index can be split up and worked on in chunks, as opposed to the entirety of the data needing to be loaded in a sparse index.

2. Query 1: Find records with a search-key value of 11
   Query 2: Find records with a search-key value between 7 and 17, inclusive.

   a. Query 1: search the first level, follow the first pointer, search the next level, follow the third pointer, search the next level and find 11
      i. Compared: 19, 5, 11

      Query 2: search the first level, follow the first pointer, search next level, follow second pointer, search next level, follow second pointer, record the entries, return to previous node, follow the pointer to next leaf node, record entries
      ii. Compared:19, 5, 11, 5, 7, 11

   b. Query 1: search the first level, follow second pointer, search next level and find 11
      i. Compared: 7, 19, 7, 11

      Query 2: Search the first level, follow the second pointer, search the second level and record entries
      ii. Compared: 7, 19, 7, 11, 17

   c. Query 1: search the first level, follow the second pointer, search the second level and find 11
      i. Compared: 11, 11

      Query 2: search the first level, follow the first pointer, search the second level and record entries, return to previous level, follow the second pointer, search second level and record entries.
      ii. Compared: 11, 2, 3, 5, 7, 11, 17

4. 17.12
List the ACID properties. Explain the usefulness of each.

Atomicity: without this there will be inconsistencies in the database
Consistency: ensures the database is in a consistent state regardless of what happens in and to it
Isolation: ensures each transaction does not affect a different transaction while running at the same time
Durability: ensures no loss of data in the database

5. 17.13
During its execution, a transaction passes through several states, until it finally commits or aborts. List all possible sequences of states through which a transaction may pass. Explain why each state transition may occur

Possible states that a transaction may be in are: Active, partially committed, committed, failed, and aborted.
Active > partially committed > committed
This is the normal sequence of events for a transaction that is successful

Active > failed > aborted
This series will occur when the transaction starts to execute, realizing it cannot execute under normal conditions. It then moves to the failed state, gets rolled back and goes into the aborted state.

Active > partially committed > failed > aborted
This series executes all statements and is moved to the partially committed state, however, before it can begin recovering a failure happens somewhere which then moves it to the failed state, and finally to the aborted state.