Homework Assignment 6

Due: 11:59PM, June 20, 2025

1.

1. (1 pt. each) Read the textbook Chapters 13, 14, and 5. Fill in the blanks with the correct answers.
(a) To acquire fast random access to tuples in a file, one can use a/an ( ) structure.
(b) An ( ) stores the values of the search-keys in sorted order and associates each search-key with the records that contain it.
(c) A ( ) is an index whose search key also defines the sequential order of the file.
(d) In a ( ), an index entry appears for only some of the search-key values.
(e) A ( ) is a statement that the system executes automatically as a side effect of a modification to the database.
(f) The SQL standard supports functions returning tables as results: such functions are called ( ).

Answer 1.

a) a) index
b) b) ordered index
c) c) clustering index
d) d) sparse index
e) e) trigger
f) f) table function

2.

(a) (3 pt.) Give at least three example items that metadata or system catalog stores.

Answer 2(a).

1. **Table information**: The names and data types of attributes in each relation.
2. **Relationships**: Referential constraints between tables (e.g., foreign key dependencies).
3. **Constraints**: Integrity constraints such as primary keys, unique constraints, and checks on attributes.

(b) (3 pt.) What are the main disadvantages of the index-sequential file organization?

Answer 2(b).

1. **Overflow pages and performance degradation**:
   When data is inserted or deleted, it may lead to overflow pages if there is no space in the sequence. These overflow pages are not in the same memory location as the original data, which leads to **non-sequential access and deteriorates performance as the number of insertions increases**.
2. **Periodic reorganization required**:
   The structure of an index-sequential file is static. As a result, **space can be wasted and performance can degrade over time**. To maintain efficiency, periodic reorganization of the entire file is needed, which **incurs a high maintenance cost**.

3.

(a) (3 pt.; Exercise 14.16) When is it preferable to use a dense index rather than a sparse index? Explain your answer.

Answer:

A dense index is preferable when the data records are **not stored in sorted order** on the search key. This is because a **dense index** maintains an index entry for **every record,** allowing efficient search even when the data is unordered.

In contrast, a **sparse index** only indexes **some of the records**, so if the **data is not sorted**, it may require a **linear scan** to locate certain records, which reduces performance.

(b) (3 pt.; Exercise 14.17) What is the difference between a clustering index and a secondary index?

Answer:

The main difference is how the index relates to the **physical order of records** on disk.

- A **clustering index** determines the **physical order** of records in the table.
  It means that the table is physically sorted based on the clustering index key. Only **one clustering index** can exist per table.

- A **secondary index**, on the other hand, does **not affect** the physical ordering of the records.
  Multiple secondary indexes can be created on different attributes.

(c) (3 pt.; Exercise 14.1) Indexes speed query processing, but it is usually a bad idea to create indexes on every attribute, and every combination of attributes, that are potentially search keys. Explain why.

Answer:

Creating indexes on every attribute and every possible combination is inefficient because:

1. **High storage overhead**: Each index requires additional storage space.

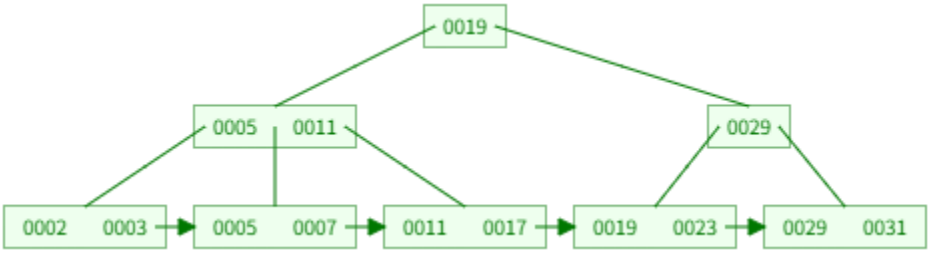
2. **Performance cost on updates**: Every time data is inserted, deleted, or updated, all associated indexes must also be updated, which slows down performance.

3. **Redundancy**: Many attributes are not frequently used as search keys, so their indexes provide little to no benefit.

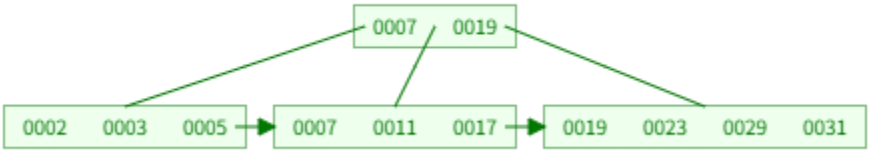
Therefore, indexes should be created selectively based on the **query workload and access patterns**.

(d) (4 pt. each; Exercise 14.3) 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 vases where the number of pointers that will fit in one nodes is as follows:
i. Four
ii. Six
iii. Eight

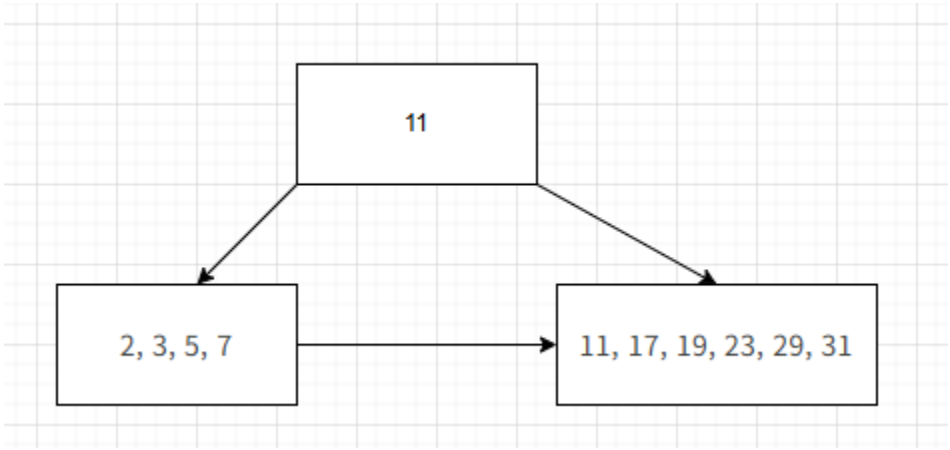
Answer i) n = 4:



Answer ii) n = 6:



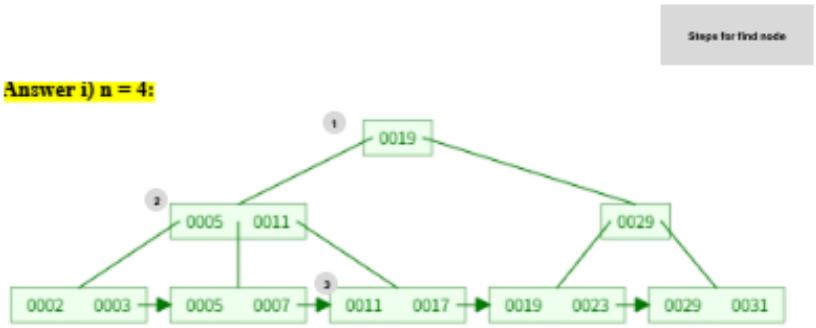
Answer iii) n = 8:



3.

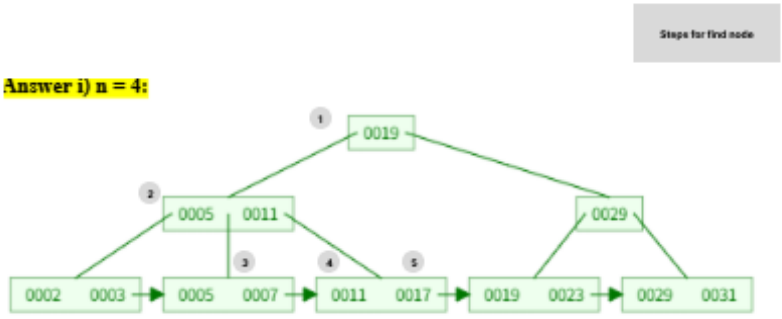
(e) (3 pt. each; Exercise 14.18) For each B+tree of Exercise 14.3a (not b and c), who the steps involved in the following queries:
i. Find records with a search-key value of 11.
ii. Find records with a search-key value between 7 and 17, inclusive.

Answer i) Point Query:



[#expain find node flow]
Start at the root [19], move to the child node [5, 11], and then to the third child leaf node [11, 17] to find key 11.

Answer ii) Point Query:

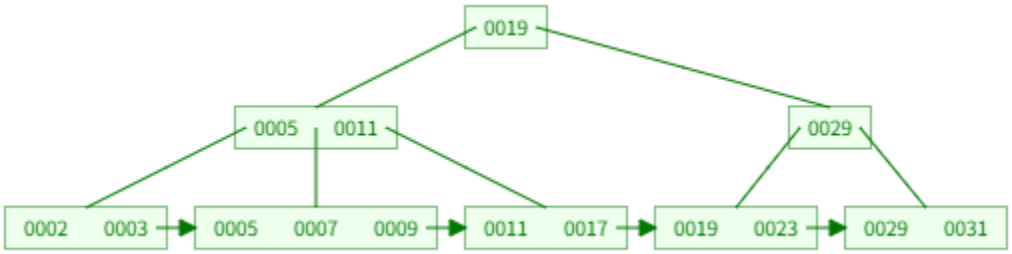


[#expain find node flow]
Start at the root [19], go to the internal node [5, 11], then sequentially scan the leaf nodes [5, 7] and [11, 17] to find keys 7 through 17.
Start at the root [19], move to the child node [5, 11], and then to the third child leaf node [11, 17] to find key 11.

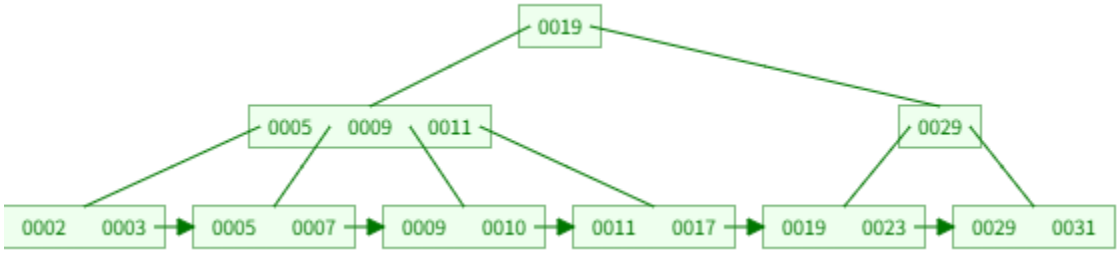
3.
(f) (5 pt. each; Exercise 14.4) For each B+tree of Exercise 14.3, show the form of the tree after each of the following series of operations:
a. Insert 9.
b. Insert 10.
c. Insert 8.

Answer i) n = 4:

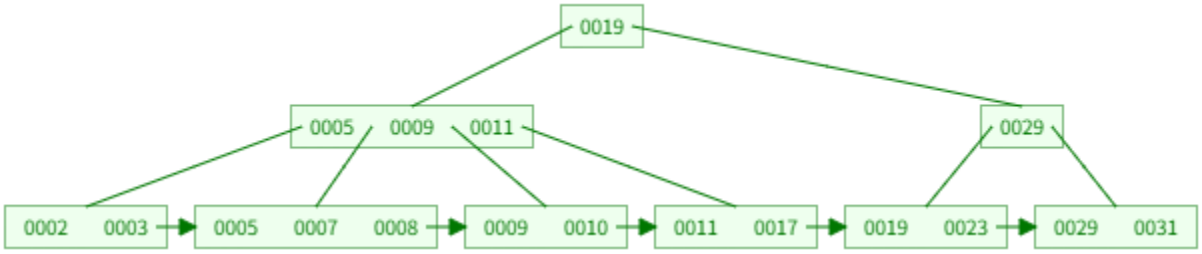
a. **Insert 9**



b. **Insert 10**



c. **Insert 8**

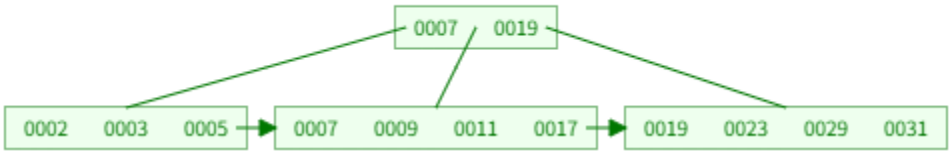


3.

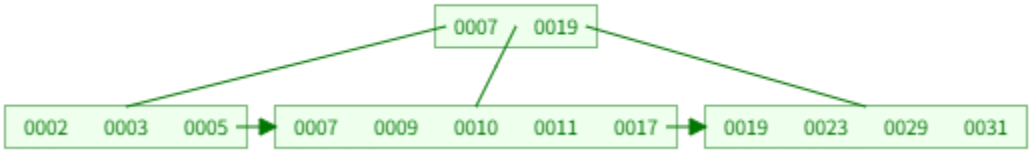
(f) (5 pt. each; Exercise 14.4) For each B+tree of Exercise 14.3, show the form of the tree after each of the following series of operations:

Answer ii) n = 6:

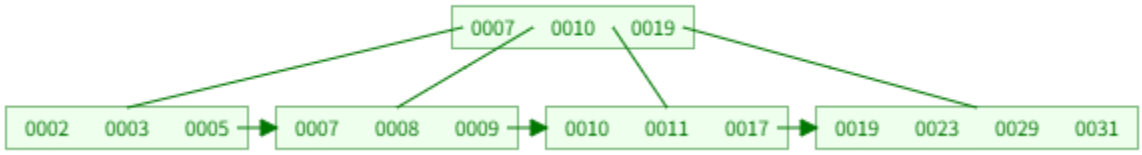
a. **Insert 9**



b. **Insert 10**



c. **Insert 8**

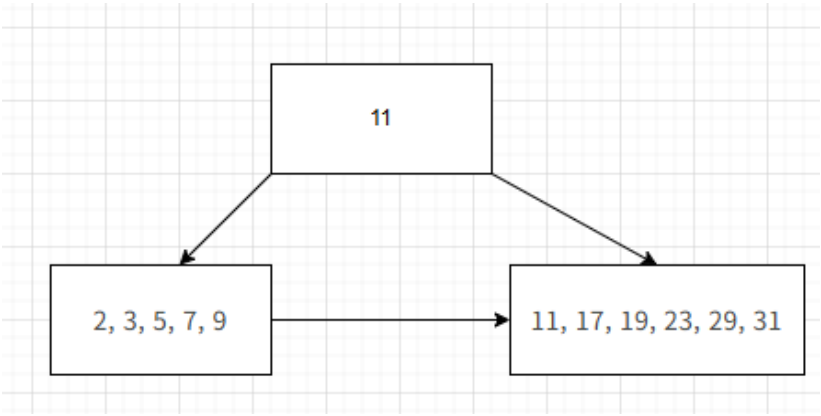


3.

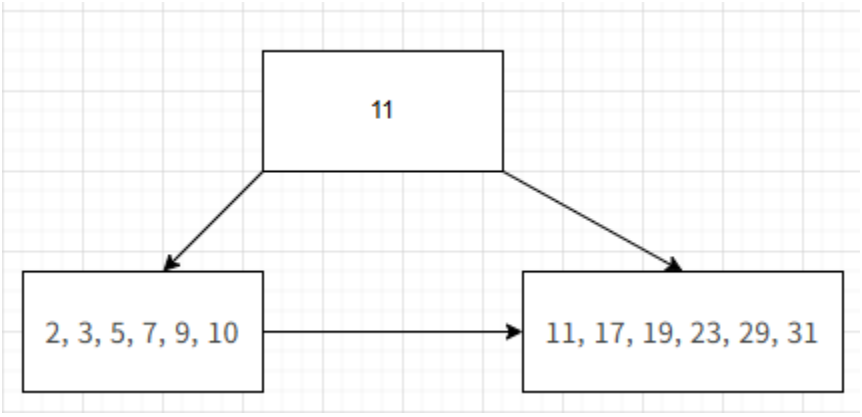
(f) (5 pt. each; Exercise 14.4) For each B+tree of Exercise 14.3, show the form of the tree after each of the following series of operations:

Answer iii) n = 8:

a. **Insert 9**



b. **Insert 10**



c. **Insert 8**