Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

CHAI
Humanities-Centered AI

Databases

Malte Luttermann – Institute for Humanities-Centered Artificial Intelligence (CHAI)

December 11, 2025

Version from December 10, 2025

Topics for Today

- Indexing
  - ISAM index structure
  - B^+ trees

Acknowledgement

- Some of the upcoming content is taken from the following lecture (and corresponding exercises) and has been translated and partially modified:
  - PD Dr. habil. Özgür L. Özçep: »Datenbanken«

Indexing – Comprehension Questions

Are the following statements correct?

- »Indexes can be used to avoid a linear scan on the database.«
- »The Indexed Sequential Access Method (ISAM) ensures that overflow pages are sorted next to the overflowing pages.«
- »Search operations are always faster in a B^+ tree than with ISAM indexing.«

Indexing – Comprehension Questions

Are the following statements correct?

- »Indexes can be used to avoid a linear scan on the database.«
  - Correct. An index can be created on any combination of attributes and maintains a logical sorting of the tuples.
- »The Indexed Sequential Access Method (ISAM) ensures that overflow pages are sorted next to the overflowing pages.«
- »Search operations are always faster in a B^+ tree than with ISAM indexing.«

Indexing – Comprehension Questions

Are the following statements correct?

- »Indexes can be used to avoid a linear scan on the database.«
  - Correct. An index can be created on any combination of attributes and maintains a logical sorting of the tuples.
- »The Indexed Sequential Access Method (ISAM) ensures that overflow pages are sorted next to the overflowing pages.«
  - Incorrect. Overflow pages are appended to the end and are not sorted. This disrupts the sequential order and causes the index to degenerate.
- »Search operations are always faster in a B^+ tree than with ISAM indexing.«

Indexing – Comprehension Questions

Are the following statements correct?

- »Indexes can be used to avoid a linear scan on the database.«
  - Correct. An index can be created on any combination of attributes and maintains a logical sorting of the tuples.
- »The Indexed Sequential Access Method (ISAM) ensures that overflow pages are sorted next to the overflowing pages.«
  - Incorrect. Overflow pages are appended to the end and are not sorted. This disrupts the sequential order and causes the index to degenerate.
- »Search operations are always faster in a B^+ tree than with ISAM indexing.«
  - Incorrect. For static and sorted data, search operations can be roughly as fast as in a B^+ tree, but performance declines when updating the data.

Indexing – Why not just Index Everything?

In an example table *Student*(*StudentId*, *FirstName*, *LastName*, *Gender*, *Address*, *PhoneNumber*), every attribute could in principle be of interest for a query, and therefore every query should be able to be processed as quickly as possible.
Why not simply index all combinations of all attributes?

Indexing – Why not just Index Everything?

In an example table *Student*(*StudentId*, *FirstName*, *LastName*, *Gender*, *Address*, *PhoneNumber*), every attribute could in principle be of interest for a query, and therefore every query should be able to be processed as quickly as possible.
Why not simply index all combinations of all attributes?

- Each index requires additional space on the disk
- An index must be maintained every time the source table is changed
  - Every time an `INSERT`, `UPDATE` or `DELETE` is performed in the source table, every index belonging to the table must be updated.

Indexing – Why not just Index Everything?

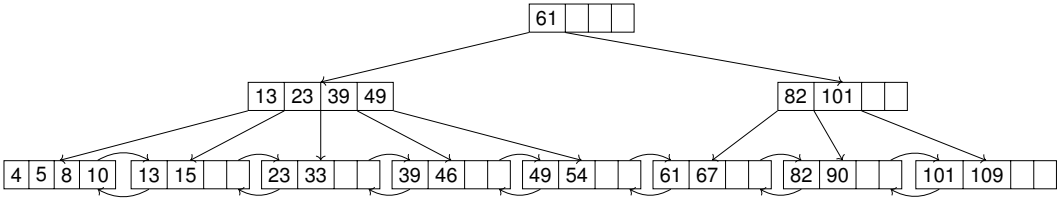
In an example table *Student*(*StudentId*, *FirstName*, *LastName*, *Gender*, *Address*, *PhoneNumber*), every attribute could in principle be of interest for a query, and therefore every query should be able to be processed as quickly as possible.
Why not simply index all combinations of all attributes?

- Each index requires additional space on the disk
- An index must be maintained every time the source table is changed
  - Every time an `INSERT`, `UPDATE` or `DELETE` is performed in the source table, every index belonging to the table must be updated.

An index therefore generally improves reading time at the expense of writing time!

Operations in B$^+$ Trees (Insertion)

Consider the following B$^+$ tree:



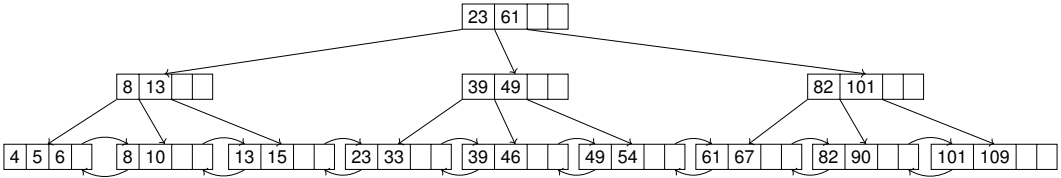
Add an entry with the key 6.

Operations in B$^+$ Trees (Insertion)

- First, the new key 6 is added
- This causes an overflow on the first leaf node $(4, 5, 6, 8, 10)$
  - Split $(4, 5, 6, 8, 10)$ into $(4, 5, 6)$ and $(8, 10)$
  - Alternatively, split into $(4, 5)$ and $(6, 8, 10)$ also possible
- The information about the split is returned to the level above
  - This creates an overflow on $(8, 13, 23, 39, 49)$
  - Split $(8, 13, 23, 39, 49)$ into $(8, 13)$ and $(39, 49)$
  - The 23 is forwarded to the next level (as a new separator)
- In the root node, 23 is added as a new separator

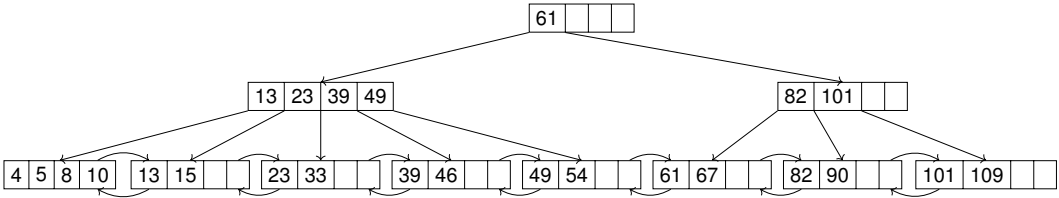
Operations in B$^+$ Trees (Insertion)

Resulting B$^+$ tree:



Operations in B$^+$ Trees (Insertion)

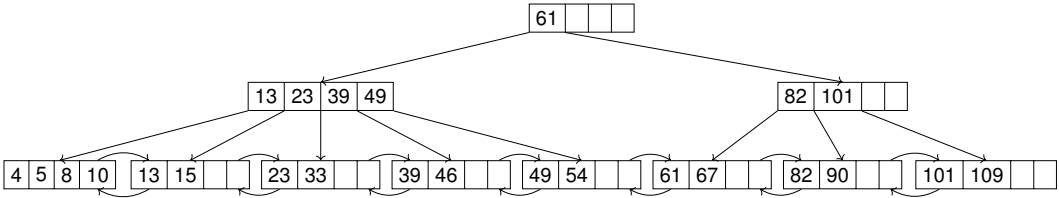
Consider the following B$^+$ tree:



How many read and write operations are required to add an entry with the key 6?

Operations in B$^+$ Trees (Insertion)

Consider the following B$^+$ tree:

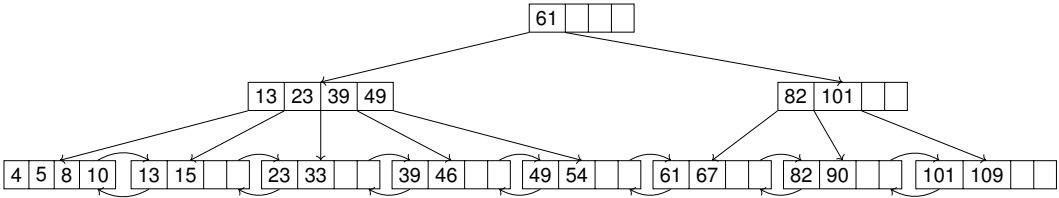


How many read and write operations are required to add an entry with the key 6?

- To find the correct node to insert, three pages must be read: (61), (13, 23, 39, 49), and (4, 5, 8, 10)
- All modified or newly created pages must be written, so 5 write accesses are required: (4, 5, 6), (8, 10), (8, 13), (39, 49), and the root page (23, 61)

Operations in B$^+$ Trees (Deletion)

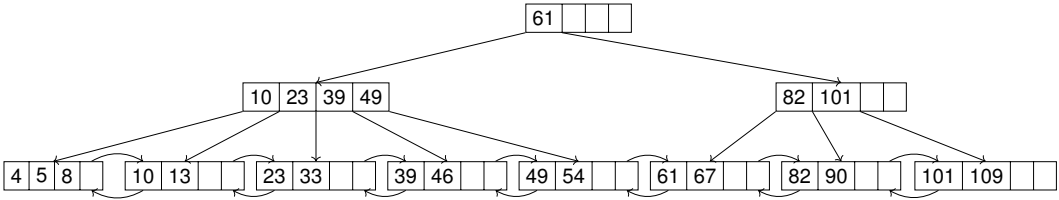
Consider the following B$^+$ tree:



Delete the entry with the key 15.

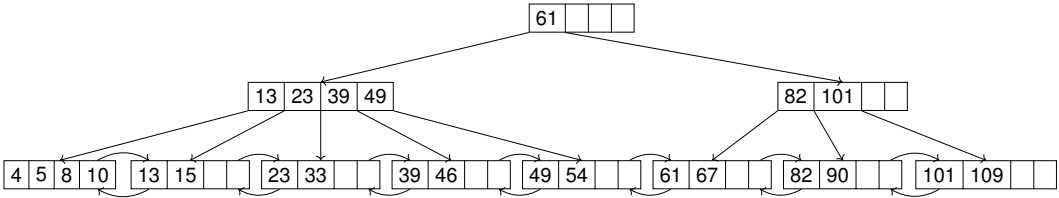
Operations in B$^+$ Trees (Deletion)

Resulting B$^+$ tree:



Operations in B$^+$ Trees (Insertion and Deletion)

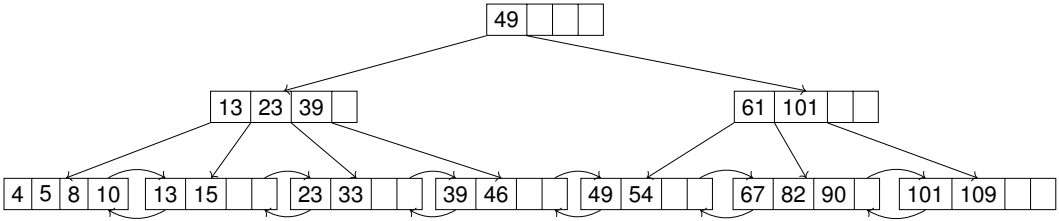
Consider the following B$^+$ tree:



Delete the entry with the key 61 and then add an entry with the key 66.

Operations in B$^+$ Trees (Insertion and Deletion)

After deleting the entry with key 61:



Operations in B$^+$ Trees (Insertion and Deletion)

Resulting B$^+$ tree: