Engineering Databases

Lecture 9 – Indexing & NoSQL Databases

January 25, 2023

M. Saeed Mafipour & Mansour Mehranfar

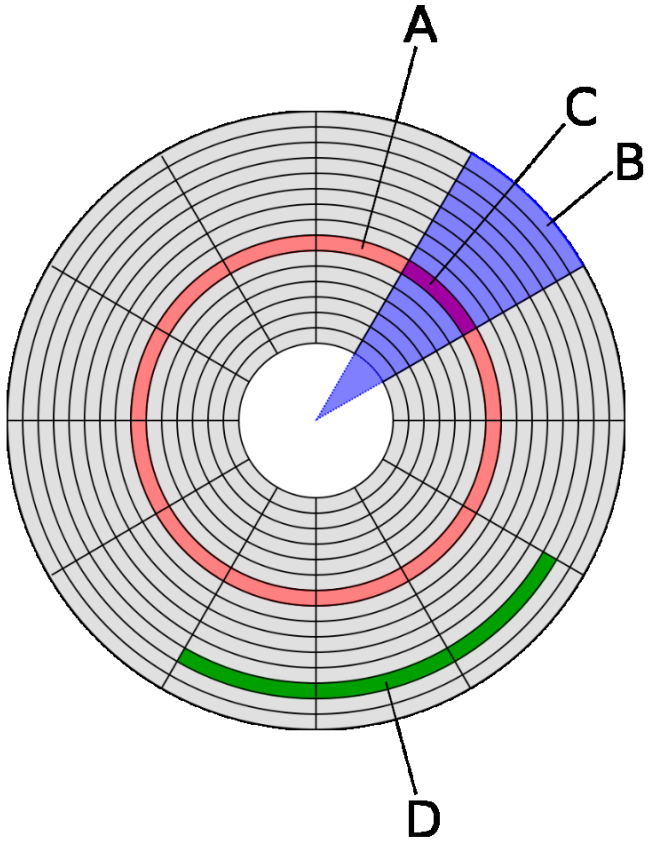
Content of Lecture 8

- Indexing improves the query performance of a database
- We measure performance based on the worst case complexity of an algorithm
- We describe the complexity via the Big-O Notation, e.g. O(1) = constant-time

- Indexing can be applied to attributes of a table
- Typically, the database creates an index for each primary key
- Index are not for free, e.g. higher effort for insert and update
- ISAM (Index Sequence Access Method) is an indexing method
- B-Tree aims for balancing the tree height or depth
- Each node and leaf is pointing to a data row

Structure of a disk

- A: Track
- B: Sector
- C: Block



RAM:

Programs/codes

Storing tables on the disk

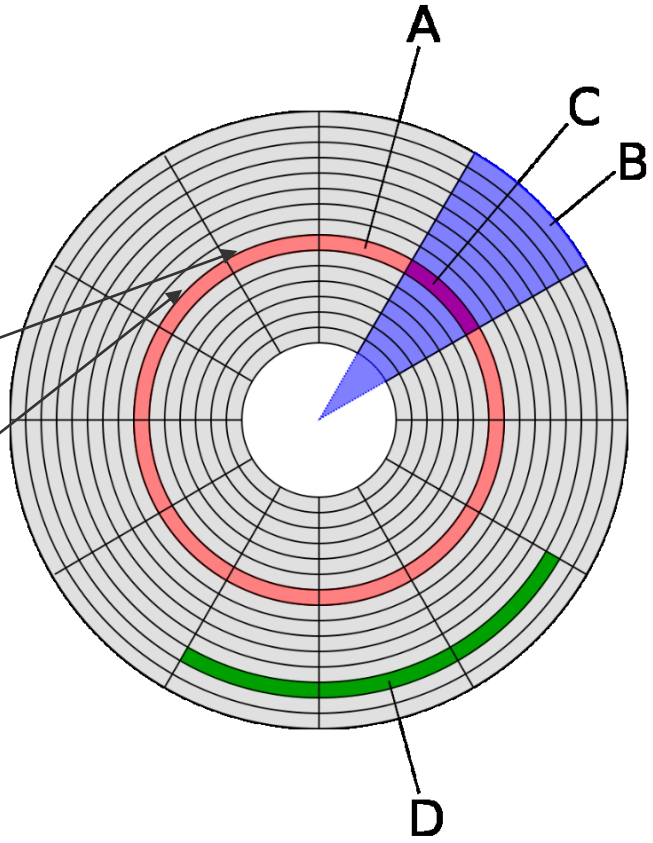
rp : reference pointer

cp : child pointer

id	cp
1	
2	
3	
4	
…	
n	

id	rp
1	
2	
3	
4	
…	
n	

Id	Name	depar tment
1	Martin	…
2	Jonas	…
3	Ann	…
4	Alex	…
…	…	…
n	…	m



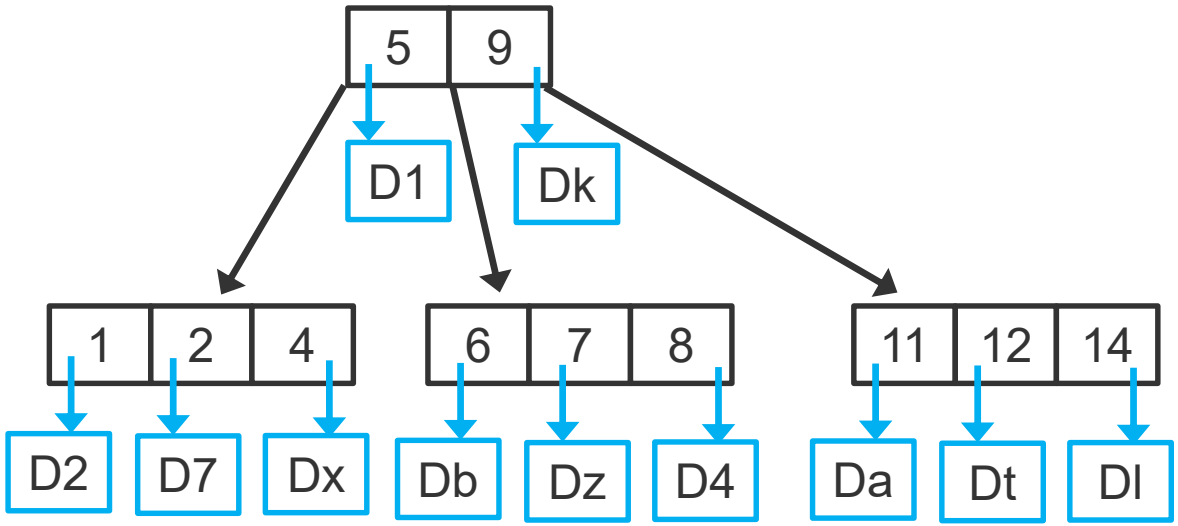
n=25
Each block takes 10 rows
25/10 = 3 blocks is needed to save the table

n=100
Each block takes 10 rows
100/10 = 10 blocks is needed to save the table

n=100
Each block takes 2 rows
100/2 = 50 blocks is needed to save the table

Indexing B-Tree Data Pointers

- In B-Trees, each key points to a data record (tuple, row).
- The real structure looks like this:



Indexing B-Tree Insert Algorithm

1. If the tree is empty,
create a root node with the key.

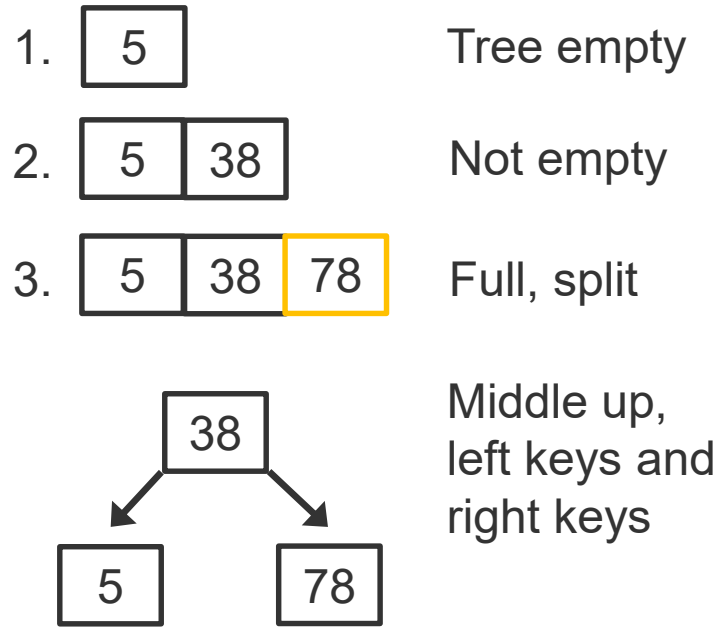
2. If the tree is not empty,
use search to find the node
where to put in the new key value.

3. If the node is not full
(maximal **m-1** keys), add the key.

4. If the node is full, split the node.

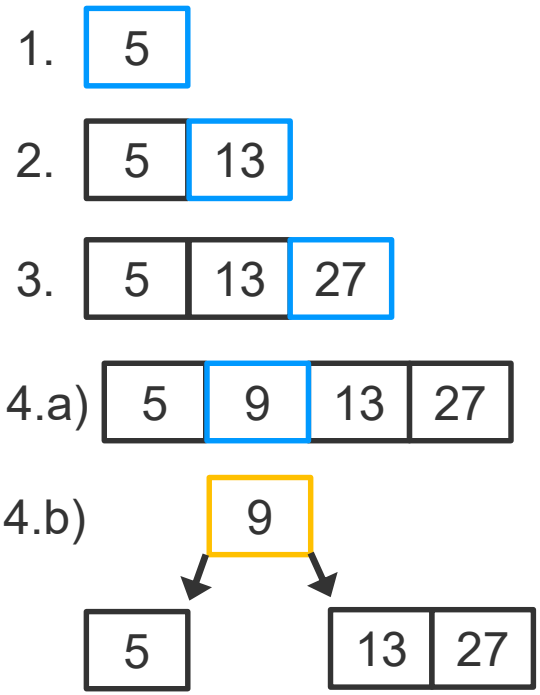
Split:
Put the middle key to the parent node.
Move other keys to new or existing nodes.
Repeat this for the parent node if it is full.

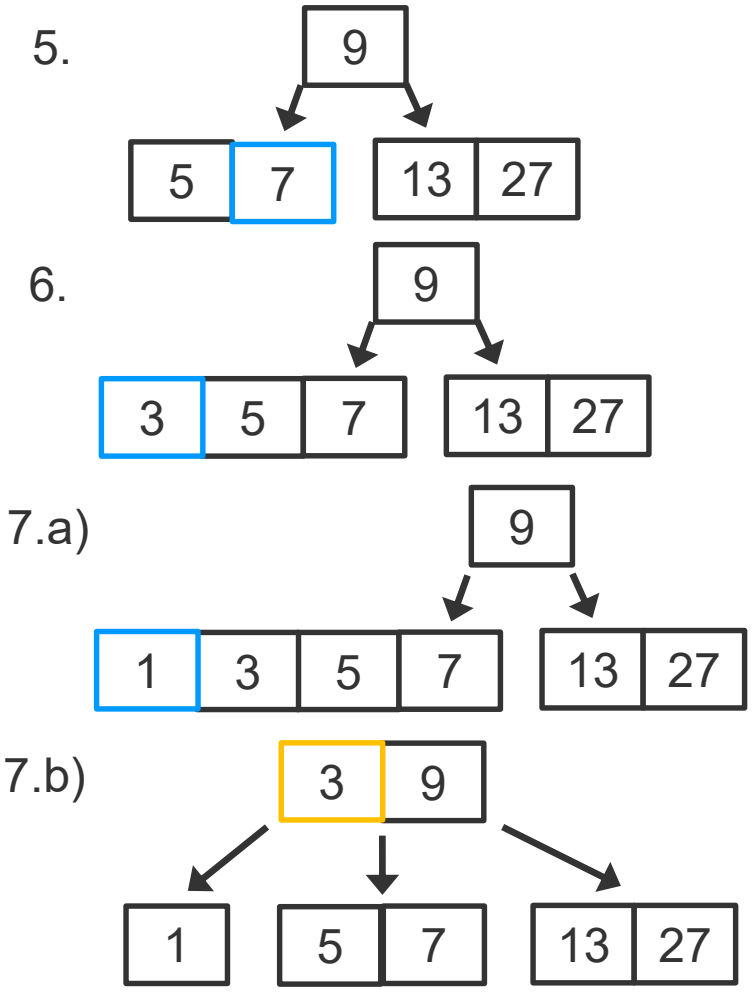
Let m=3, min: 3/2-1=1, max = m-1=2



Indexing B-Tree Insert Algorithm Exercise

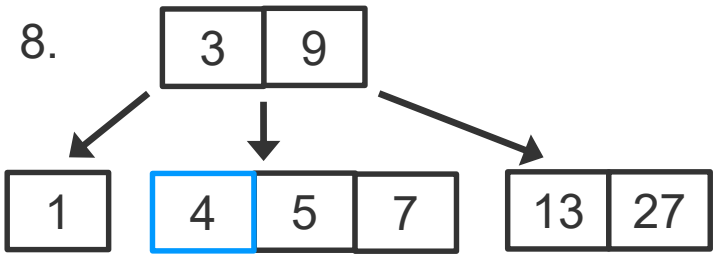
- Let m=4, min: 4/2-1=1, max = m-1=3
- Add 5, 13, 27, 9, 7, 3, 1, 4, 8, 10, 11

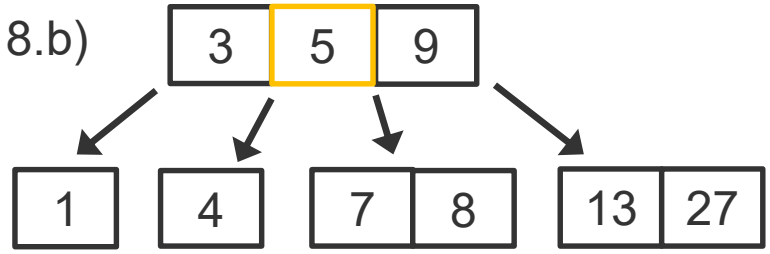


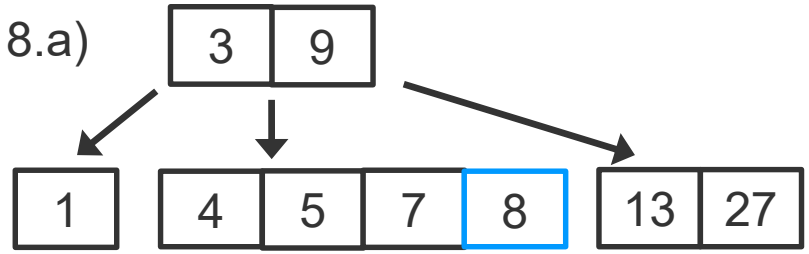


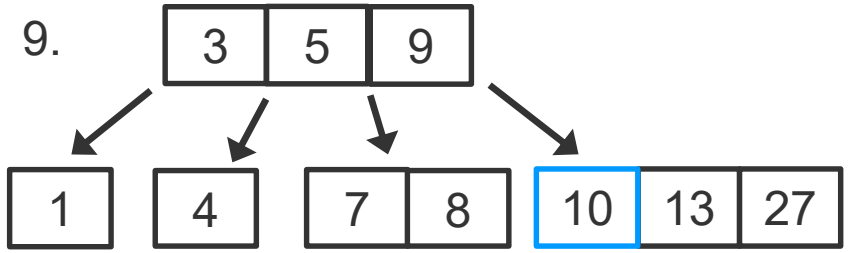
Indexing B-Tree Insert Algorithm Exercise

- Let m=4, min: 4/2-1=1, max = m-1=3
- Add 5, 13, 27, 9, 7, 3, 1, 4, 8, 10, 11





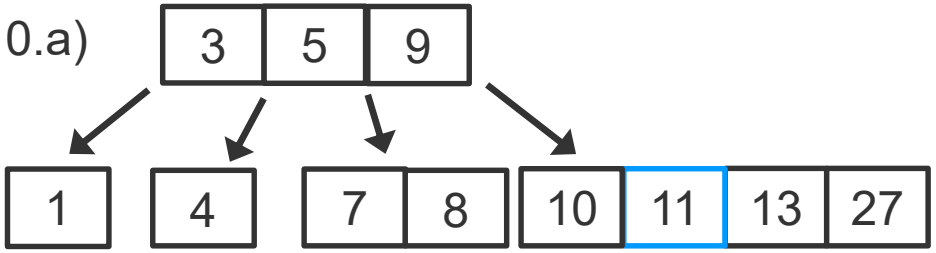




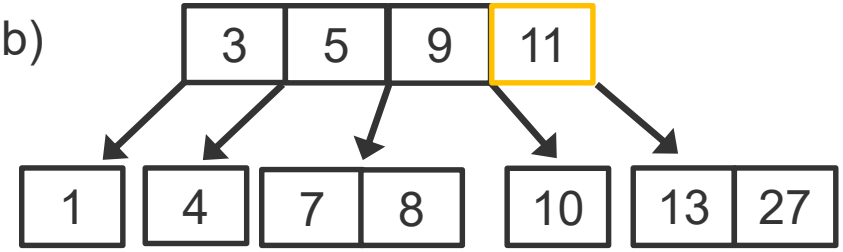
Indexing B-Tree Insert Algorithm Exercise

- Let m=4, min: 4/2-1=1, max = m-1=3
- Add 5, 13, 27, 9, 7, 3, 1, 4, 8, 10, 11

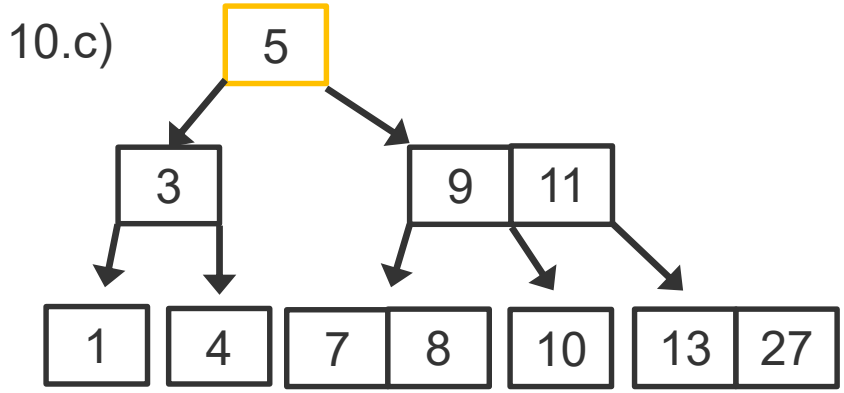
10.a)



10.b)



10.c)



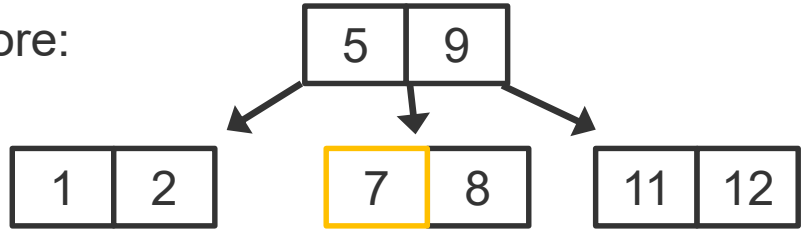
Indexing B-Tree Delete Algorithm

1. Use search to find the key in the tree. If not found, stop.

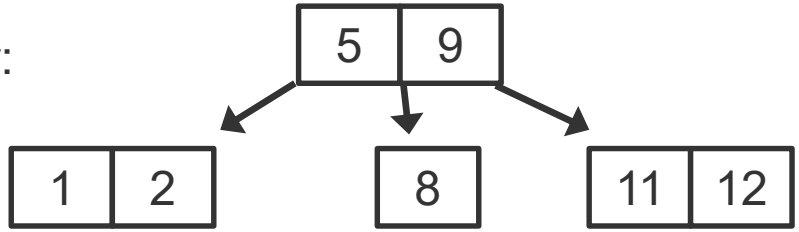
2. If the key is in a leaf, delete it if no rule is violated.

Let m=3, min: 3/2-1=1, max = m-1=2

Before:



After:



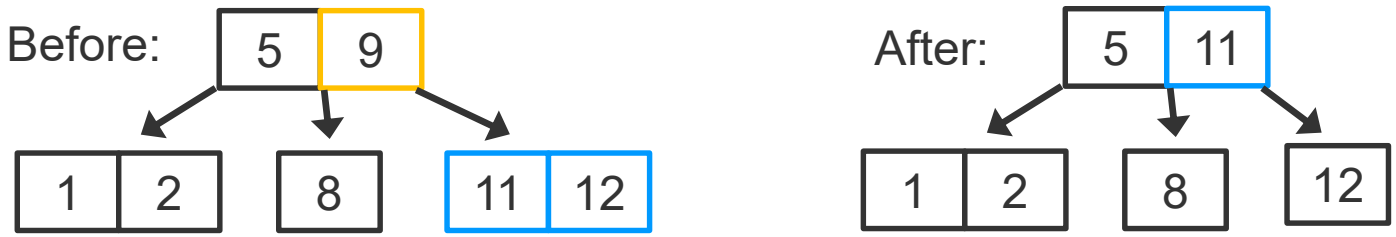
Indexing B-Tree Delete Algorithm

3. If the key is in an internal node X:

Let m=3, min: 3/2-1=1, max = m-1=2

a)
If a child node K of X has at least one more key than the minimum (m/2-1).
Delete the key in X and move an appropriate key from K to X.

The delete operation may have be done recursively.



Indexing B-Tree Delete Algorithm

3. If the key is in an internal node X:

Let m=3, min: 3/2-1=1, max = m-1=2

b)
If no child node of X has at least one more key than the minimum (m/2-1).
Merge the child nodes.

The delete operation may have be done recursively.

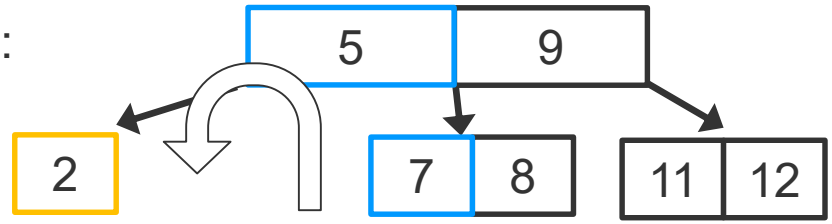
Indexing B-Tree Delete Algorithm

4. If the key is in a leaf X and deleting it would violated rules.

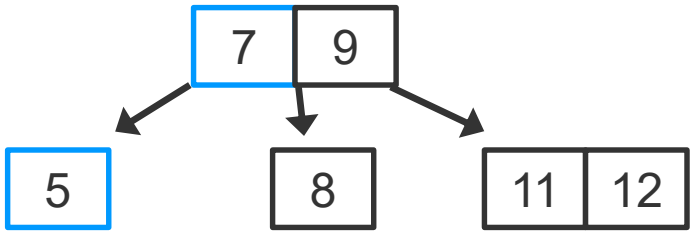
Let m=3, min: 3/2-1=1, max = m-1=2

a)
If a sibling node K has at least one more key than the minimum (m/2-1).
Rotate the keys to the left (or to the right)

Before:



After:



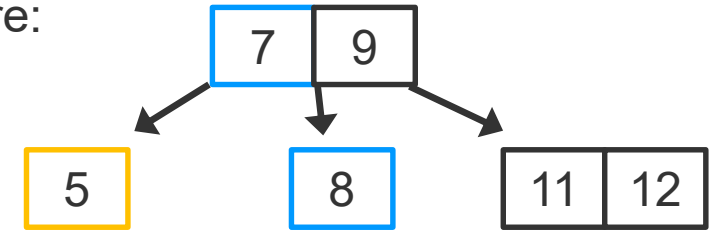
Indexing B-Tree Delete Algorithm

4. If the key is in a leaf X and deleting it would violated rules.

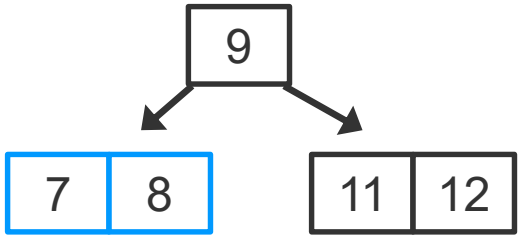
Let m=3, min: 3/2-1=1, max = m-1=2

b)
If no sibling node has at least one more key than the minimum (m/2-1) but the parent node has at least one more key than the minimum.
Merge a sibling and a key from the parent node.

Before:



After:



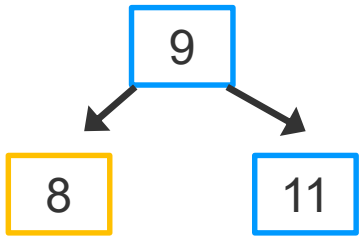
Indexing B-Tree Delete Algorithm

4. If the key is in a leaf X and deleting it would violated rules.

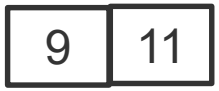
Let m=3, min: 3/2-1=1, max = m-1=2

c)
If the sibling nodes and the parent node have the minimum (m/2-1) number of keys.
Merge a sibling and the parent node (reduce the height of the tree).

Before:



After:



Indexing B-Tree Exercise

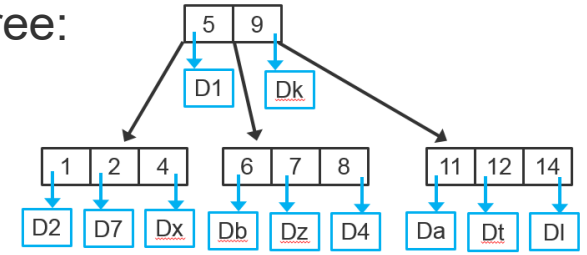
- Start with an empty tree and different m=3 (and m=4)
- Add “Apple“ (2) to the tree
- Add "Adam“ (1) to the tree
- Add "Eve“ (3) to the tree
- Add "Snake“ (6) to the tree
- Remove "Apple" (2) from the tree
- Add "Fig Leaf" (4) to the tree
- Add "Pudency“ (5) to the tree
- Remove "Adam“ (1) from the tree
- Remove "Eve" (3) from the tree

Helpful animation at:
https://www.cs.usfca.edu/~galles/visualization/BTree.html

Max. Degree =
maximal number of child nodes
Max. degree = m

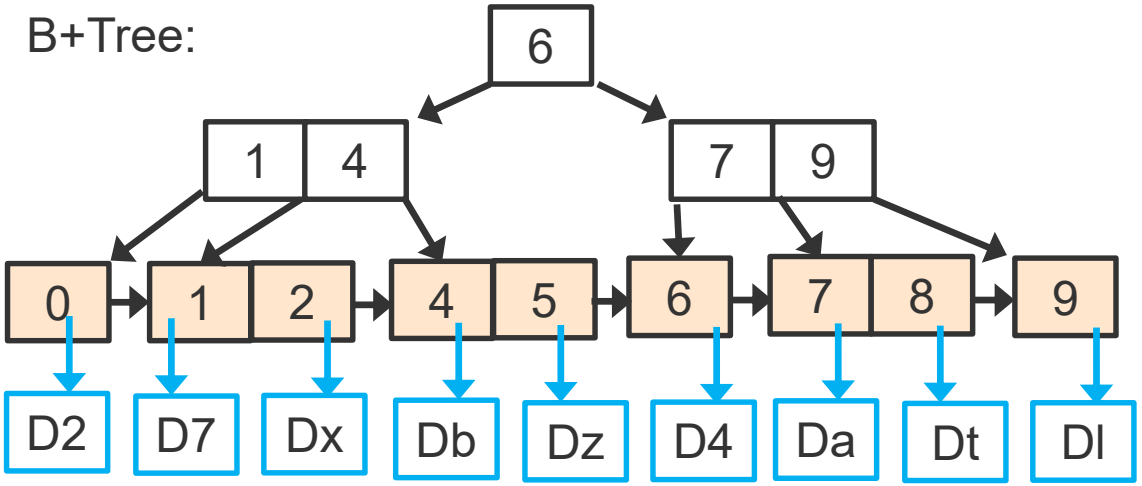
Indexing B+ Trees

B-Tree:



- Extension of the B-Tree

B+Tree:



- Nodes only store keys
- The pointers in the leaves point to data
- Leaves are connected by additional pointers: fast sequential access
- B+ Trees "grow from the root“

Indexing B+ Trees

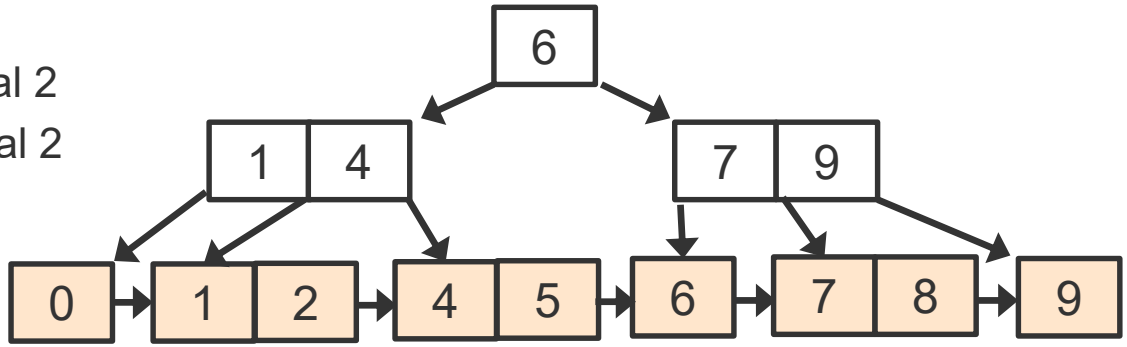
- Background information

- B+Trees are used by
- filesystems
  - ReiserFS, NSS, XFS, JFS, ReFS, BFS, NTFS, EXT4

- databases:
  - IBM DB2, Informix, Microsoft SQL Server, Oracle 8, Sybase ASE, SQLite, CouchDB, Tokyo Cabinet, MySQL (InnoDB)

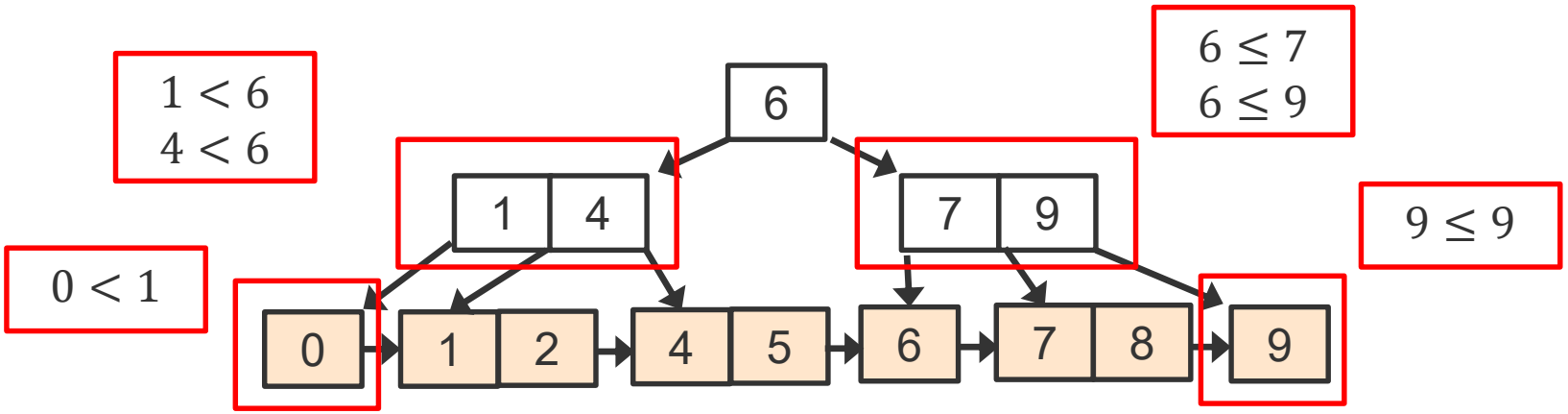
Indexing B+ Trees

- The rules for a B+Tree
  - Maximum number of keys per node: n
  - Root node: at least 2 children
  - Non-root nodes (inner nodes): at least (n+1)/2 children (rounding up, ceiling)
  - The values are sorted
  - All leaves are on the same depth
- Let n = 2
  Keys = maximal 2
  Root-node children = minimal 2
  Inner node children = minimal 2



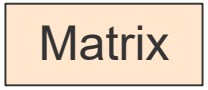
Indexing B+ Trees

- Right of key S_i are all values v with v >= S_i
- Left of key S_i are all values v with v < S_i
- The larger part goes to the right child



B+ Tree Example

- Let n = 2
- Add "Matrix"



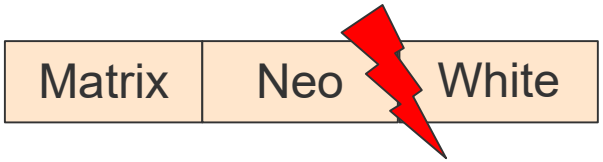
B+ Tree Example

- Let n = 2
- Add "Neo"

B+ Tree Example

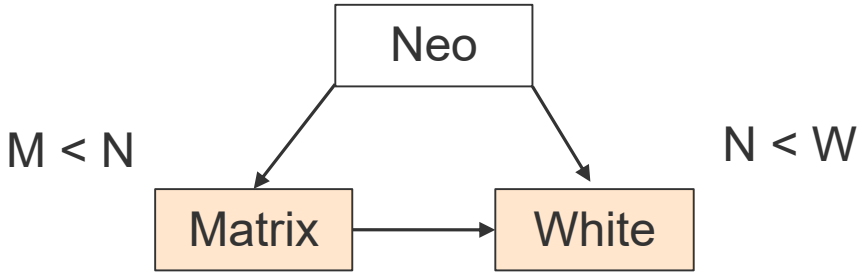
- Let n = 2
- Add „White"

Maximum number of keys per node: n=2



Split in the middle (Neo)

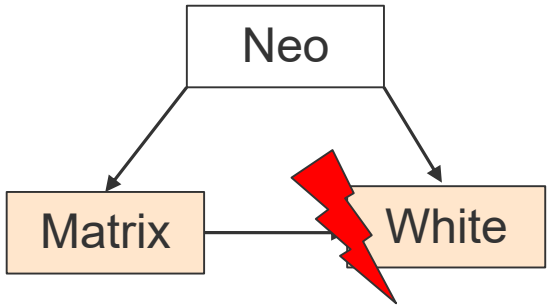
Grow to the root



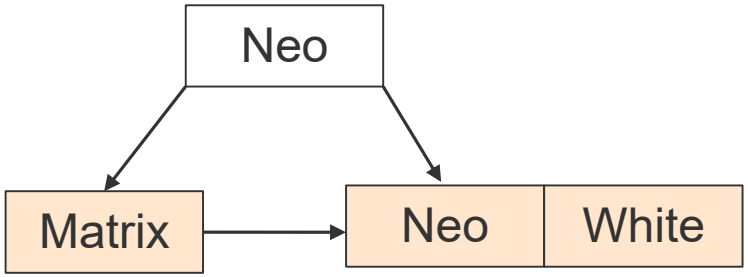
B+ Tree Example

- Let n = 2
- Add „White"

We need to have Neo in the leaves as data pointer

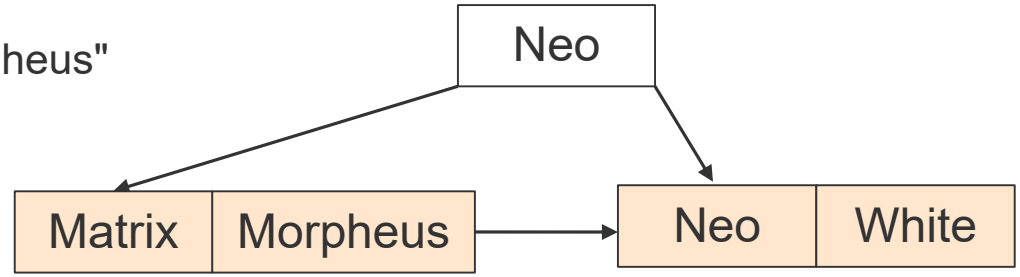


Propagate Neo to the leaf following ≤



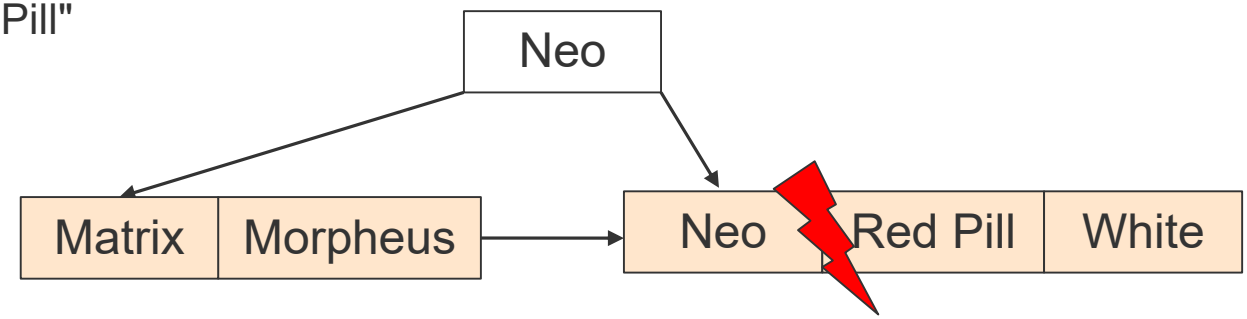
B+ Tree Example

- Let n = 2
- Add „Morpheus"

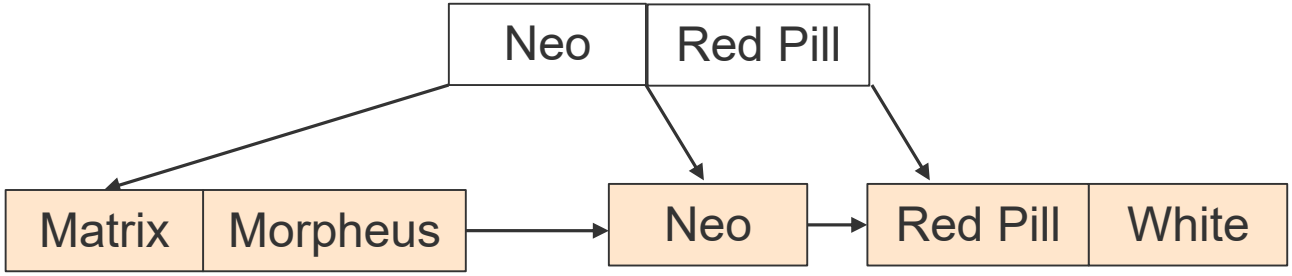


B+ Tree Example

- Let n = 2 Maximum number of keys per node: n=2
- Add „Red Pill"



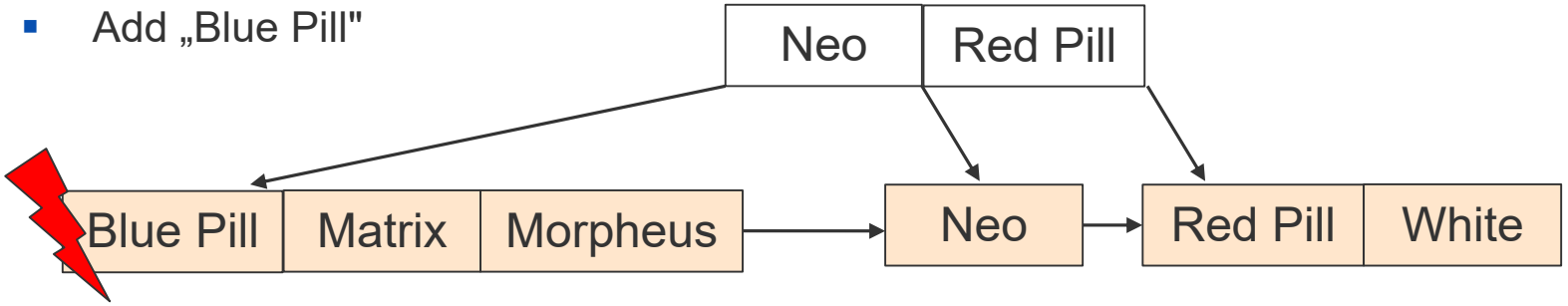
Split in the middle (Red Pill) Grow to the root



B+ Tree Example

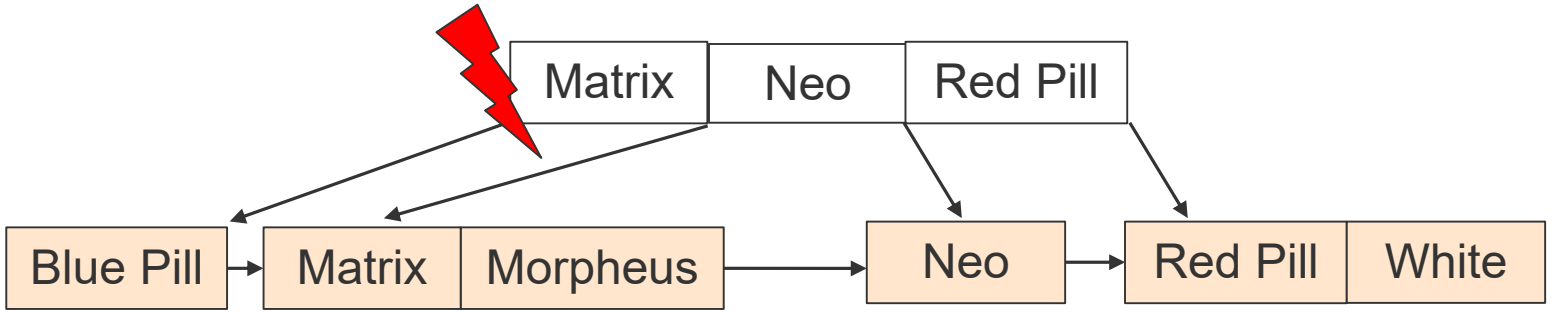
- Let n = 2
- Add „Blue Pill"

Maximum number of keys per node: n=2



Split in the middle (Blue Pill)

Grow to the root



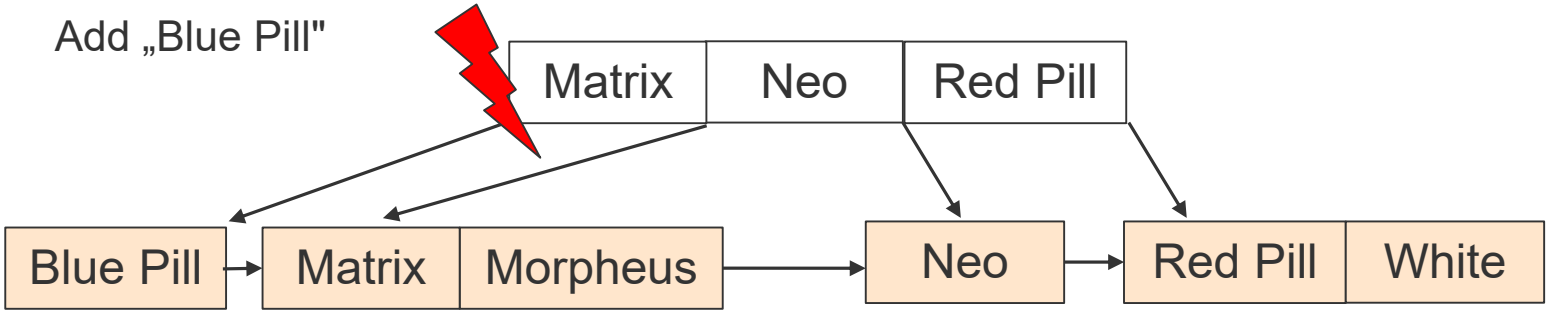
Split in the middle (Neo)

Grow to the root

B+ Tree Example

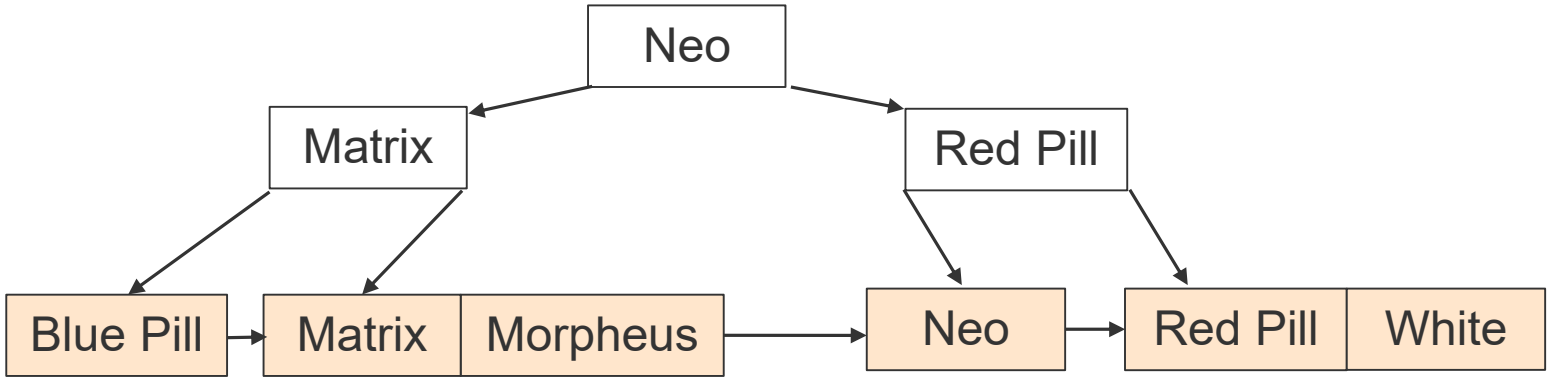
- Let n = 2
- Add „Blue Pill"

Maximum number of keys per node: n=2



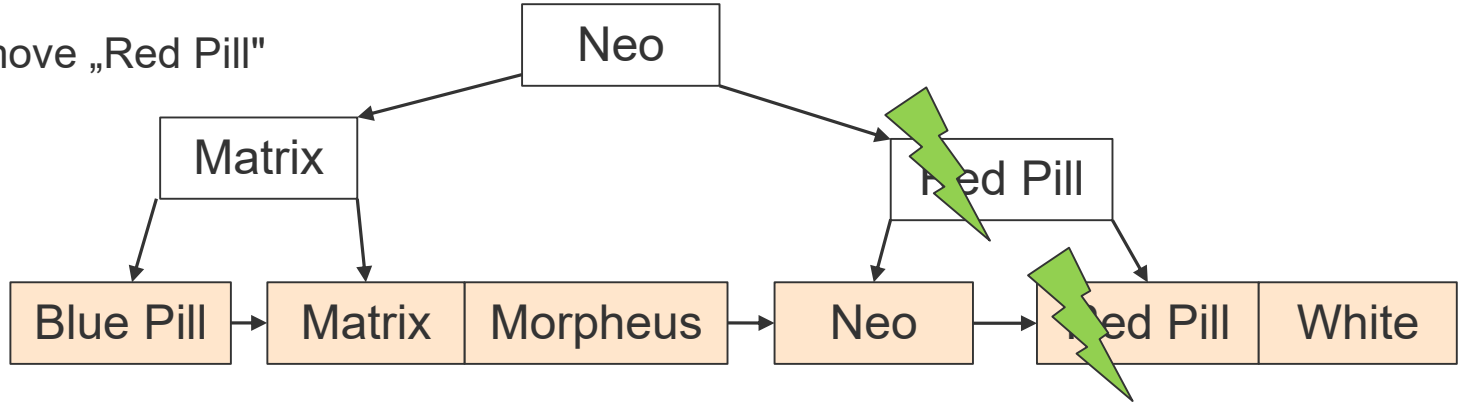
Split in the middle (Neo)

Grow to the root

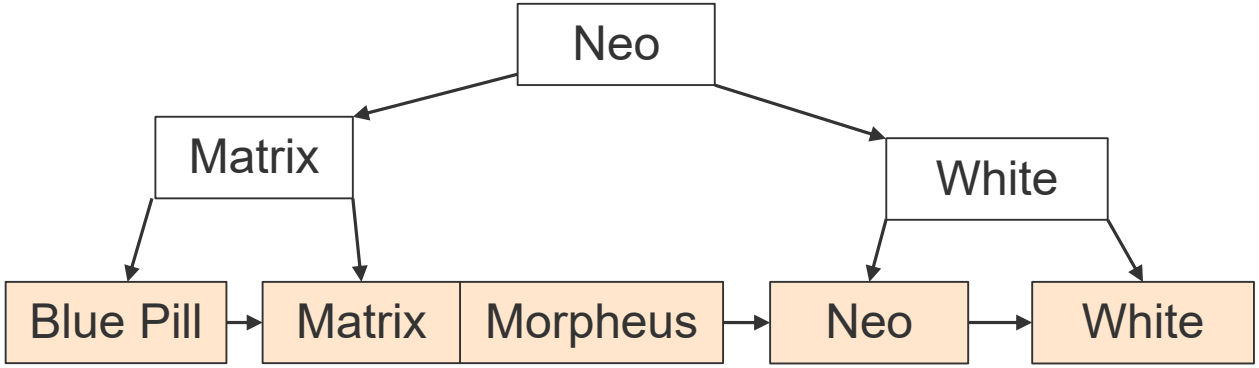


B+ Tree Example

- Let n = 2
- Remove „Red Pill"

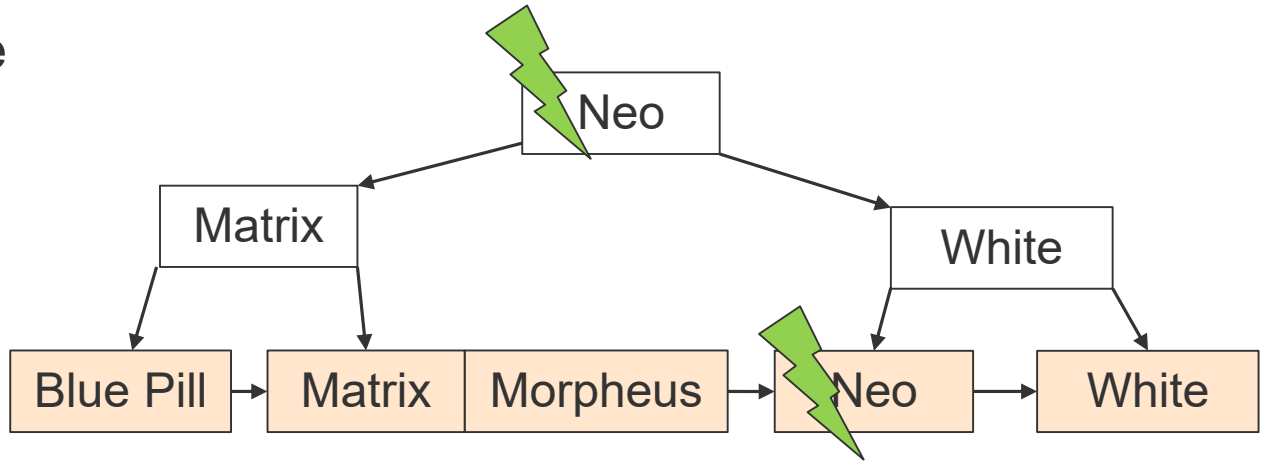


Update keys, here the next element “White” (ascending)

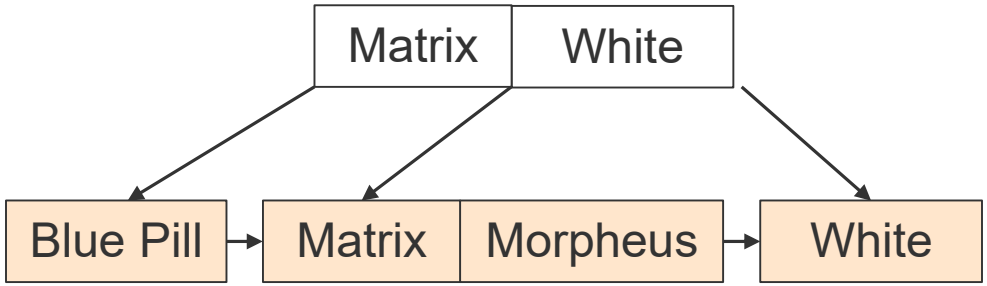


B+ Tree Example

- Let n = 2
- Remove „Neo"

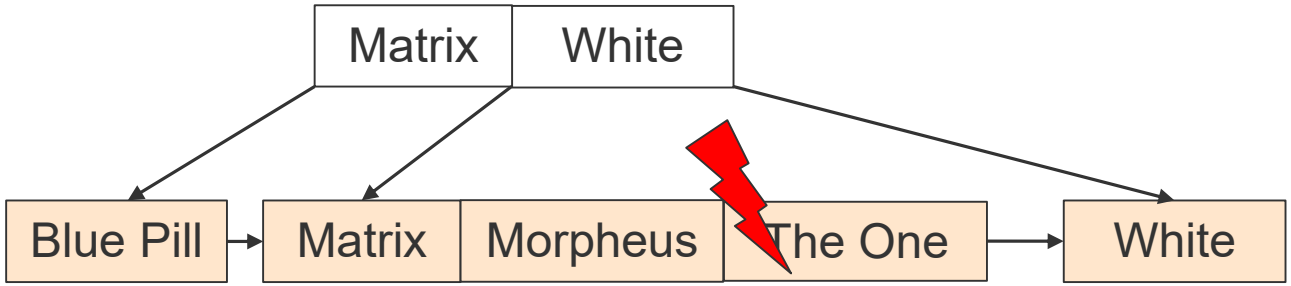


Merge children “Matrix” and “White” if parent “Neo” is gone
Connected sequence pointer if element “Neo” is gone



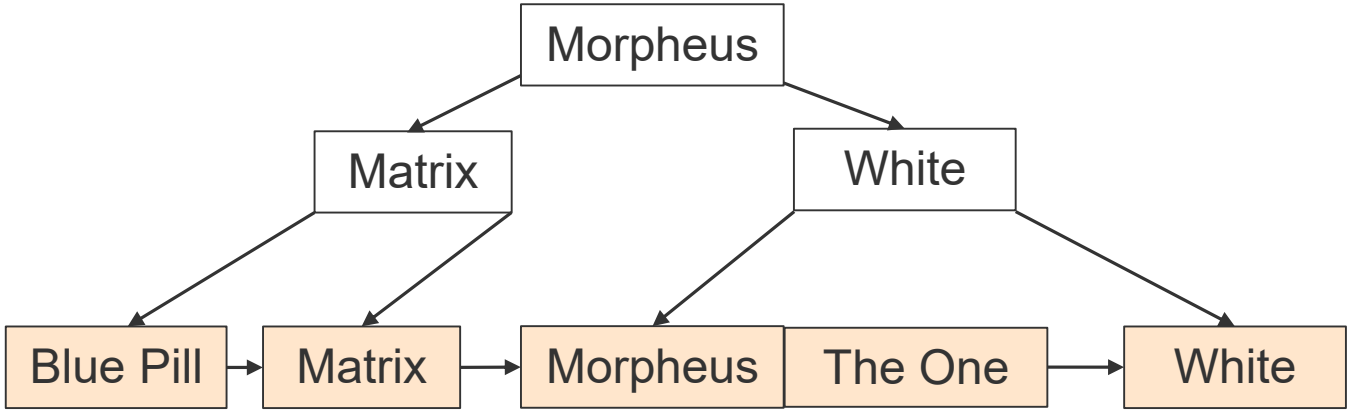
B+ Tree Example

- Let n = 2
- Add „The One"



Split in the middle (Morpheus)

Grow to the root



B+ Tree Expertise

- Start with an empty tree with n=2
- Add "Port" to the tree
- Add "Titanic"
- Add "Leo"
- Add "Kate"
- Add "Necklace"
- Add "Drama"
- Add "Painting"
- Add "Iceberg"
- Remove "Titanic" then remove "Painting"
- Add "Door"
- Add "Decision"
- Remove "Leo", then remove "Necklace"

https://www.cs.usfca.edu/~galles/visualization/BPlusTree.html

Visualization of B+Tree

Difference Between B-Tree And B+ Tree

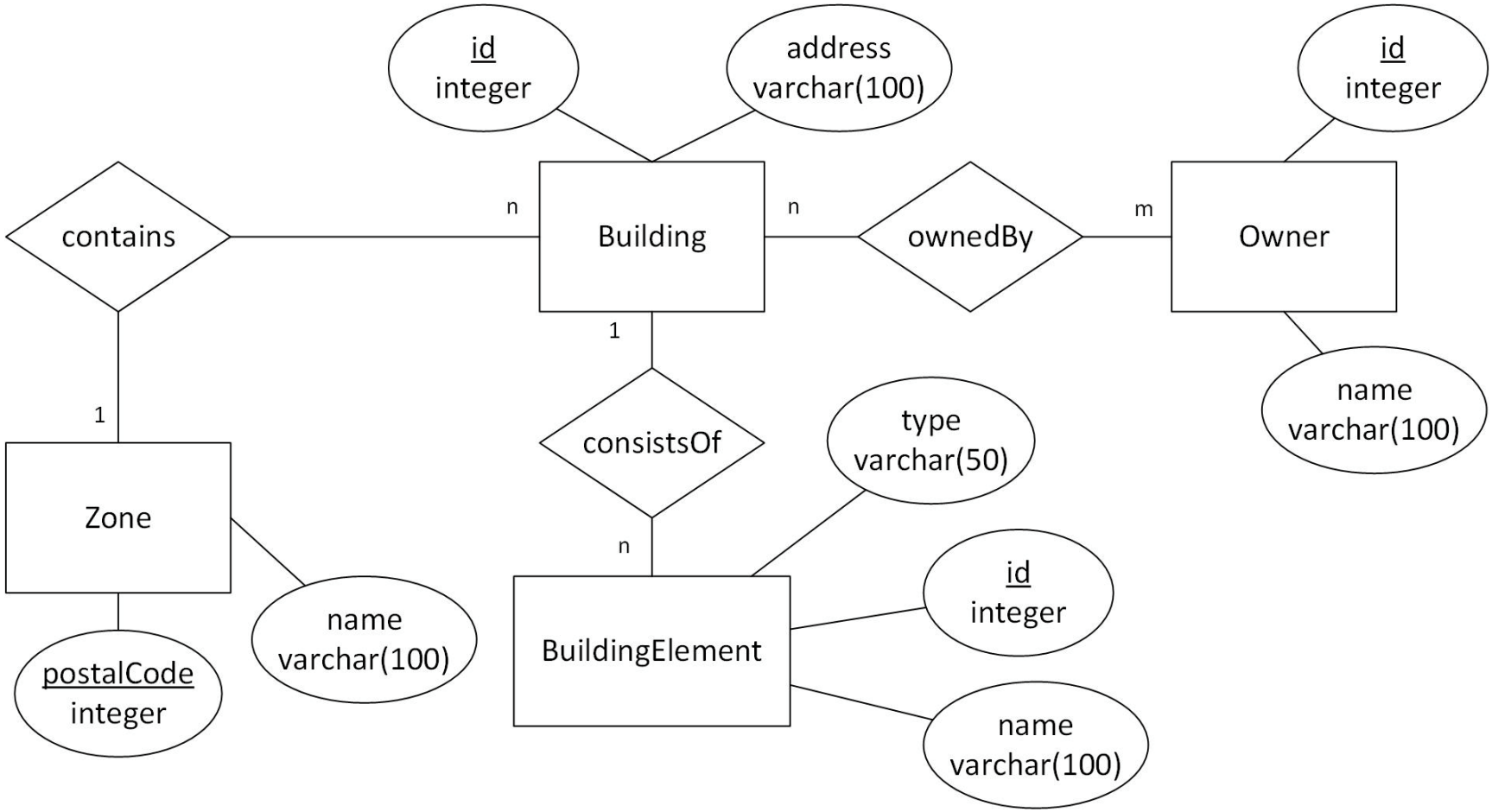
B-Tree	B+ Tree
Data is stored in leaf nodes as well as internal nodes.	Data is stored only in leaf nodes.
Searching is a bit slower as data is stored in internal as well as leaf nodes.	Searching is faster as the data is stored only in the leaf nodes.
No redundant search keys are present.	Redundant search keys may be present.
Deletion operation is complex.	Deletion operation is easy as data can be directly deleted from the leaf nodes.
Leaf nodes cannot be linked together.	Leaf nodes are linked together to form a linked list.

Table from: https://www.softwaretestinghelp.com/b-tree-data-structure-cpp/#:~:text=The%20difference%20in%20B%2B%20tree,stored%20only%20in%20internal%20nodes.&text=Internal%20nodes%20of%20the%20B%2B%20tree%20are%20called%20index%20nodes.

Assignment Remarks

- Modeling the database using ER-diagram means that you need to:
  - Draw entities, attributes, and relationships (each has its commonly used shape)
  - Mark uniqueId with an underline
  - Specify each attribute's data type: integer, varchar (40)…etc
  - Mark the cardinalities

Assignment Remarks - example



NoSQL Databases (A short Overview)

- Rise of Big Data and distributed systems

- Big Data comprise information that have a large volume, velocity, or/and variety
  - Data from e.g. Millions of users in social networks or the internet of things
  - Extract knowledge based on this large amount of information

- Fast services (e.g. web) can only be realized with many servers
  - Distributed of data
  - Cloud based computing

- New technologies were developed to solve these new challenges

NoSQL Databases

- Relational Databases
  - Are well standardized and established
  - Come with well developed technologies
  - Have concurrency control via transactions (ACID)

- Relational Databases are successful but have problems
  - Large volumes of data
  - Cloud computing
  - Velocity of data
  - Large number of users
  - Variety of data

NoSQL Databases

- NoSQL most of the time means “Not Only SQL”, sometimes “not SQL”

- NoSQL databases mostly:
  - Do not use the relation model
  - Are designed to run in large clusters
  - Do not have a schema
  - Are often highly fault-tolerant and query efficient
  - Are useful for storing unstructured data

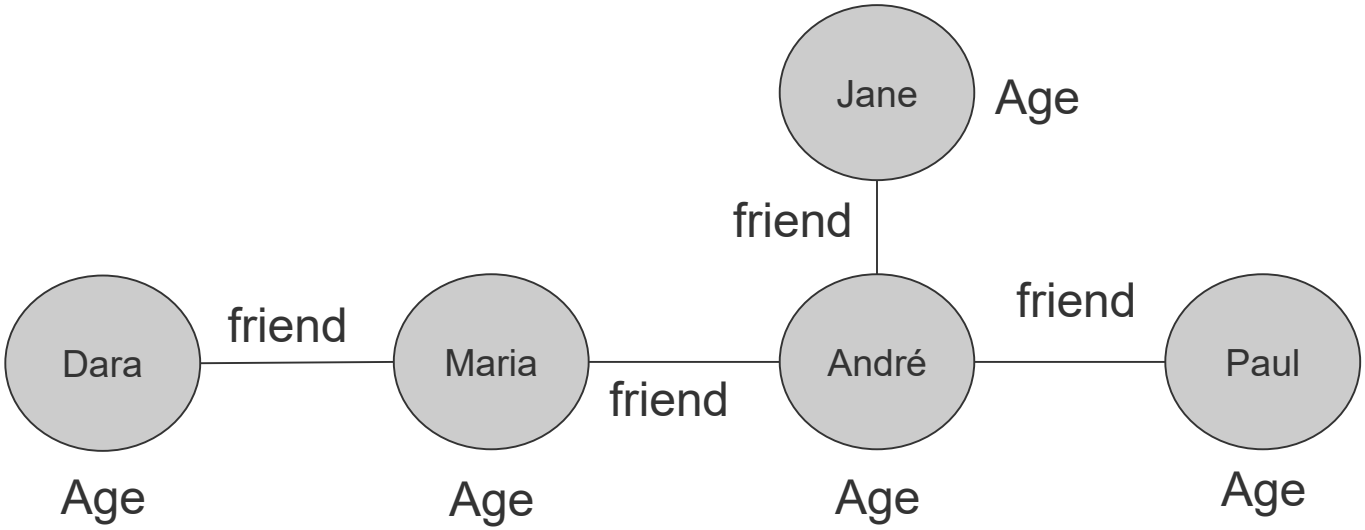
- NoSQL databases have been adopted to overcome relational databases limitations

NoSQL Databases

- NoSQL Technologies
  - Map-Reduce model
  - Key-Value stores
  - Document databases
  - Column-family stores
  - Graph databases

NoSQL Databases (A short Overview)

- Graph databases
  - Store entities and relations in form of nodes and edges
  - Nodes are instances (e.g. a student)
  - Nodes have properties (similar to attributes)
  - Edges define relations (e.g. likes)



End of Lecture

Thank you for your attention