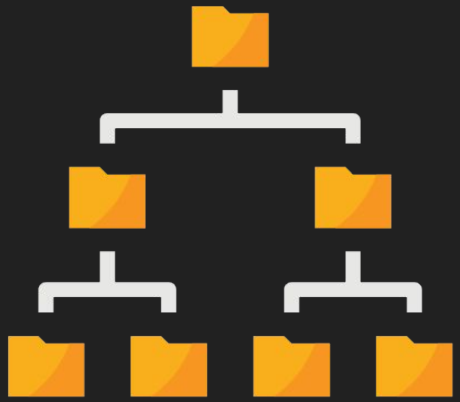


B-Trees vs B+Trees

husseinnasser.com

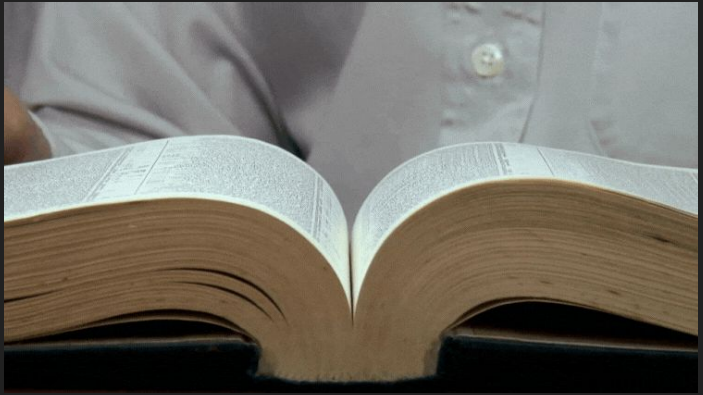
And their impact on production database systems

Agenda

- Full Table Scans
- B-Tree
- B-Tree limitations
- B+Tree
- B+Tree Considerations
- B+Tree storage cost in MySQL vs Postgres
- Summary

Full Table Scan

- To find a row in a large table we perform full table scan
- Reading large tables is slow
- Requires many I/Os to read all pages
- We need a way to reduce the search space



Find Person with ID 5

ID	,	Name
1	,	John
2	,	Ali
3	,	Rick
4	,	Sara
5	,	Edmond
….		
….		
….		
591828734	,	Chansy
591828735	,	Bird
591828736	,	Chip
591828737	,	Sud
591828738	,	Will
…..		
…..		
…..		
999991929	,	Dorit
999991930	,	Sally
999991931	,	Kory

Find Person with ID 591828738

ID , Name
1 , John
2 , Ali
3 , Rick
4 , Sara
5 , Edmond
….
….
….
591828734 , Chansy
591828735 , Bird
591828736 , Chip
591828737 , Sud
591828738 , Will
…..
…..
…..
999991929 , Dorit
999991930 , Sally
999991931 , Kory

Find Person with ID 999991931

ID	,	Name
1	,	John
2	,	Ali
3	,	Rick
4	,	Sara
5	,	Edmond
….		
….		
….		
591828734	,	Chansy
591828735	,	Bird
591828736	,	Chip
591828737	,	Sud
591828738	,	Will
…..		
…..		
…..		
999991929	,	Dorit
999991930	,	Sally
999991931	,	Kory

B-Tree

- Balanced Data structure for fast traversal
- B-Tree has Nodes
- In B-Tree of “m” degree some nodes can have (m) child nodes
- Node has up to (m-1) elements

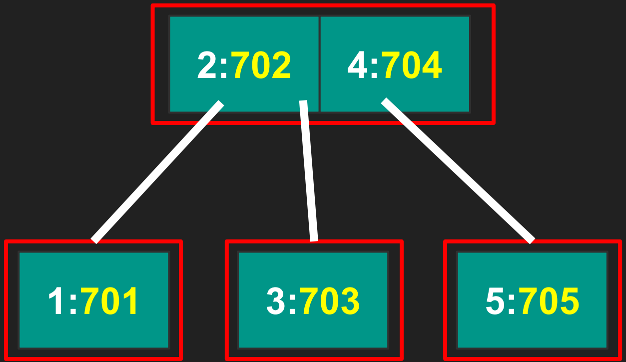
B-Tree

- Each element has a key and a value
- The value is usually data pointer to the row
- Data pointer can point to primary key or tuple
- Root Node, internal node and leaf nodes
- A node = disk page

How B-Tree Helps

TID	ID	,	Name
701	1	,	John
702	2	,	Ali
703	3	,	Rick
704	4	,	Sara
705	5	,	Edmond

Internal tuple
id/page#/rowid



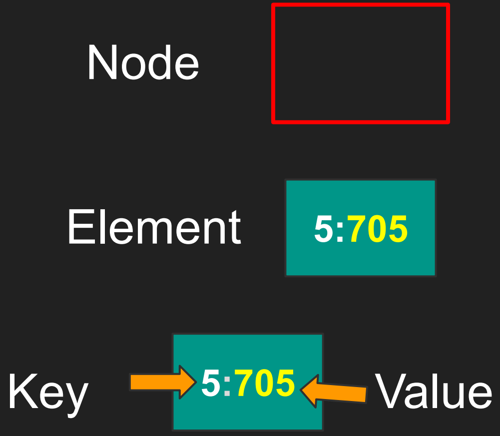


Figure 2 is an example of a B-tree in $\tau(2,3)$ satisfying all the above conditions. In the figure the α_i are not shown and the page pointers are represented graphically. The boxes represent pages and the numbers outside are page numbers to be used later.

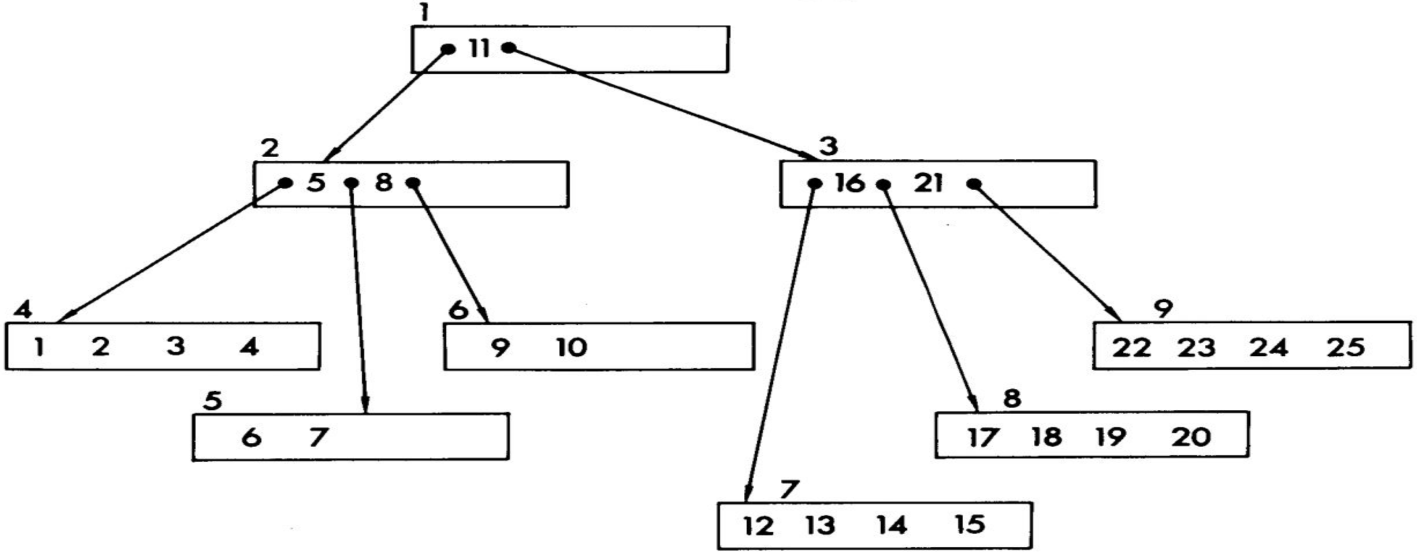


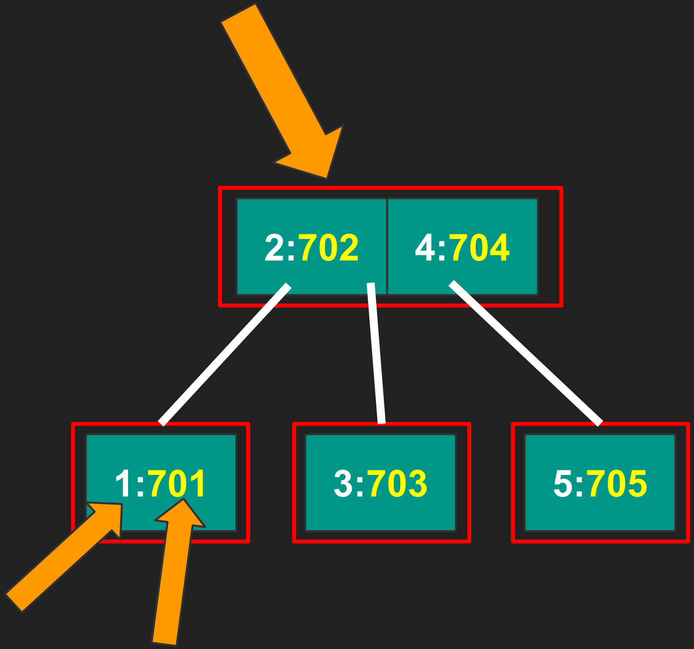
Figure 2. A Data Structure in $\tau(2,3)$ for an Index

B-Tree paper https://infolab.usc.edu/csci585/Spring2010/den_ar/indexing.pdf

Find ID 3

TID	ID	,	Name
701	1	,	John
702	2	,	Ali
703	3	,	Rick
704	4	,	Sara
705	5	,	Edmond

Find ID 1



TID	ID	,	Name
701	1	,	John
702	2	,	Ali
703	3	,	Rick
704	4	,	Sara
705	5	,	Edmond

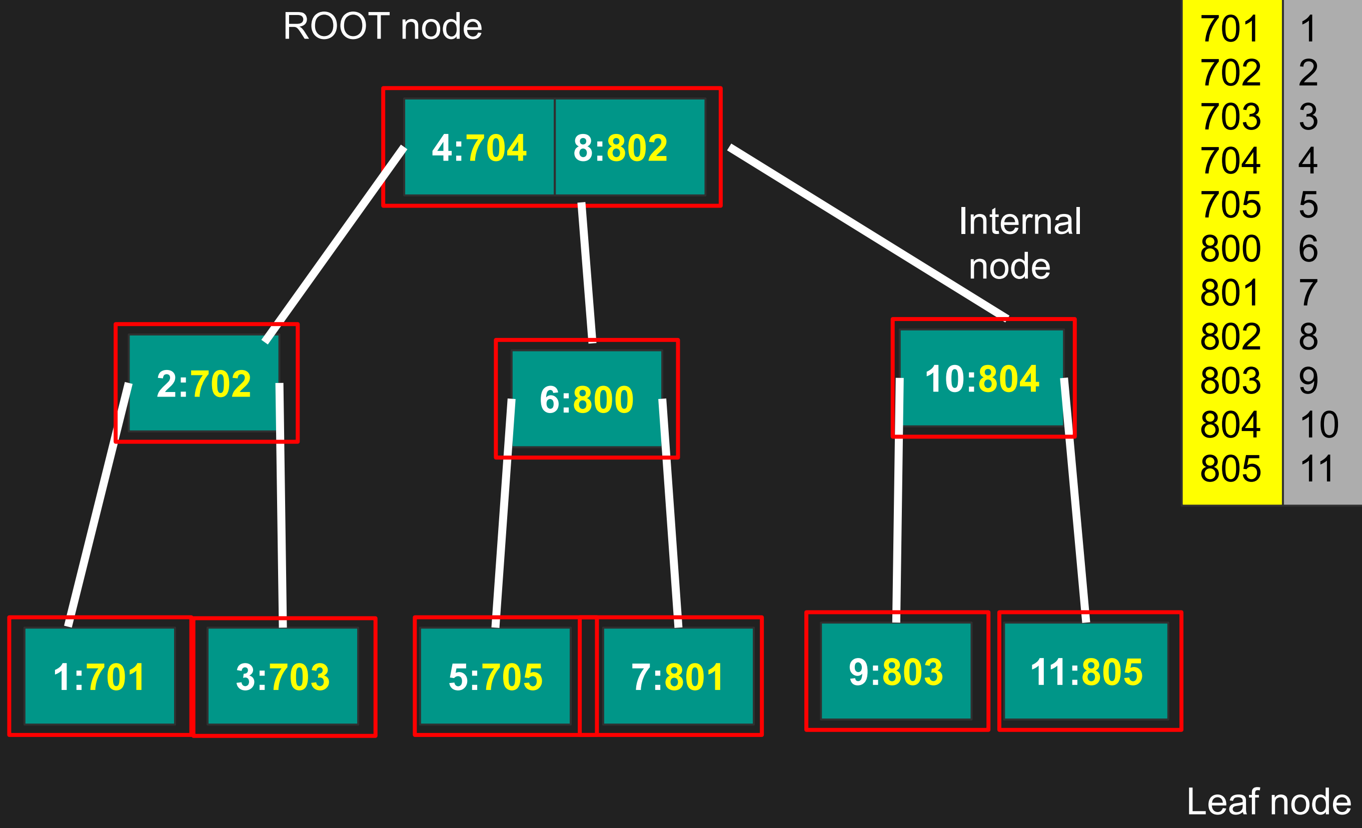
Find ID 5

TID	ID	,	Name
701	1	,	John
702	2	,	Ali
703	3	,	Rick
704	4	,	Sara
705	5	,	Edmond

Adding more entries

TID	ID	,	Name
701	1	,	John
702	2	,	Ali
703	3	,	Rick
704	4	,	Sara
705	5	,	Edmond
800	6	,	Ion
801	7	,	Edmond
802	8	,	Pete
803	9	,	Oliver
804	10	,	Yong
805	11	,	Xui

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

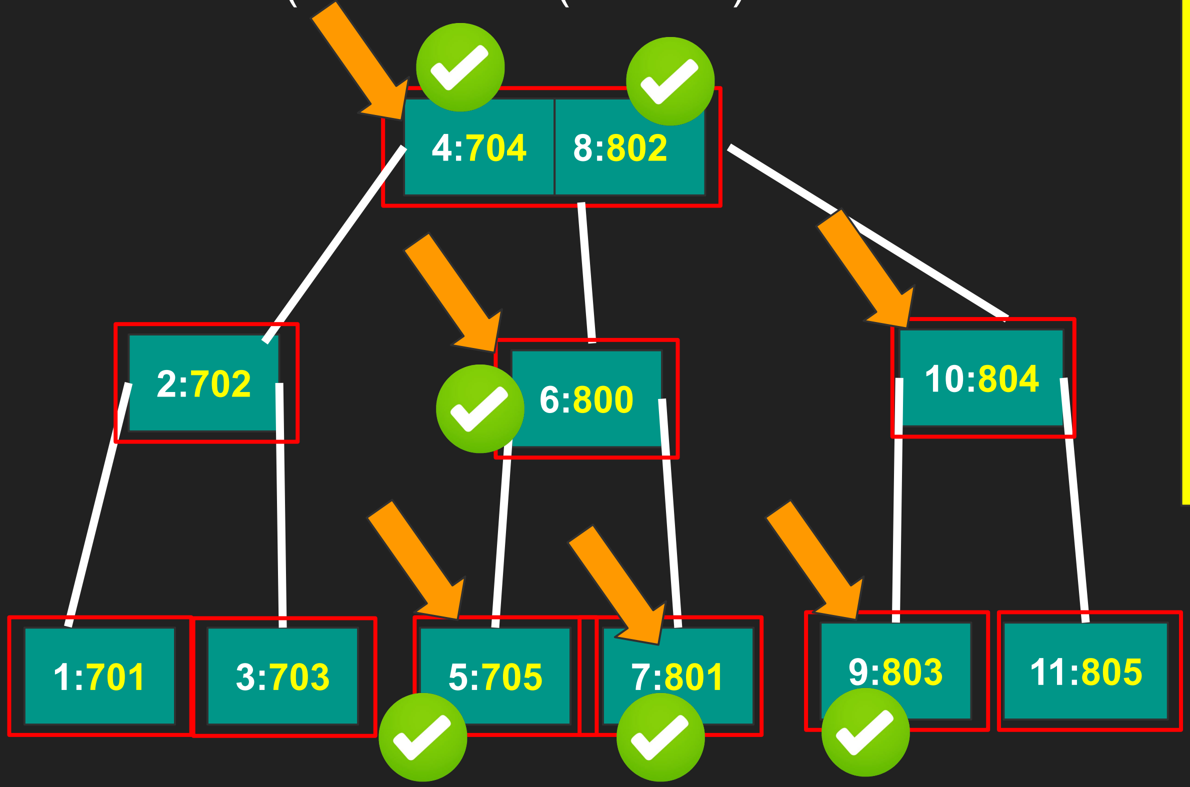


TID	ID	,	Name
701	1	,	John
702	2	,	Ali
703	3	,	Rick
704	4	,	Sara
705	5	,	Edmond
800	6	,	Ion
801	7	,	Edmond
802	8	,	Pete
803	9	,	Oliver
804	10	,	Yong
805	11	,	Xui

Limitation B-Tree

- Elements in all nodes store both the key and the value
- Internal nodes take more space thus require more IO and can slow down traversal
- Range queries are slow because of random access (give me all values 1-5)
- B+Tree solves both these problems
- Hard to fit internal nodes in memory

Find rows (ID between (4 and 9) in this b-Tree

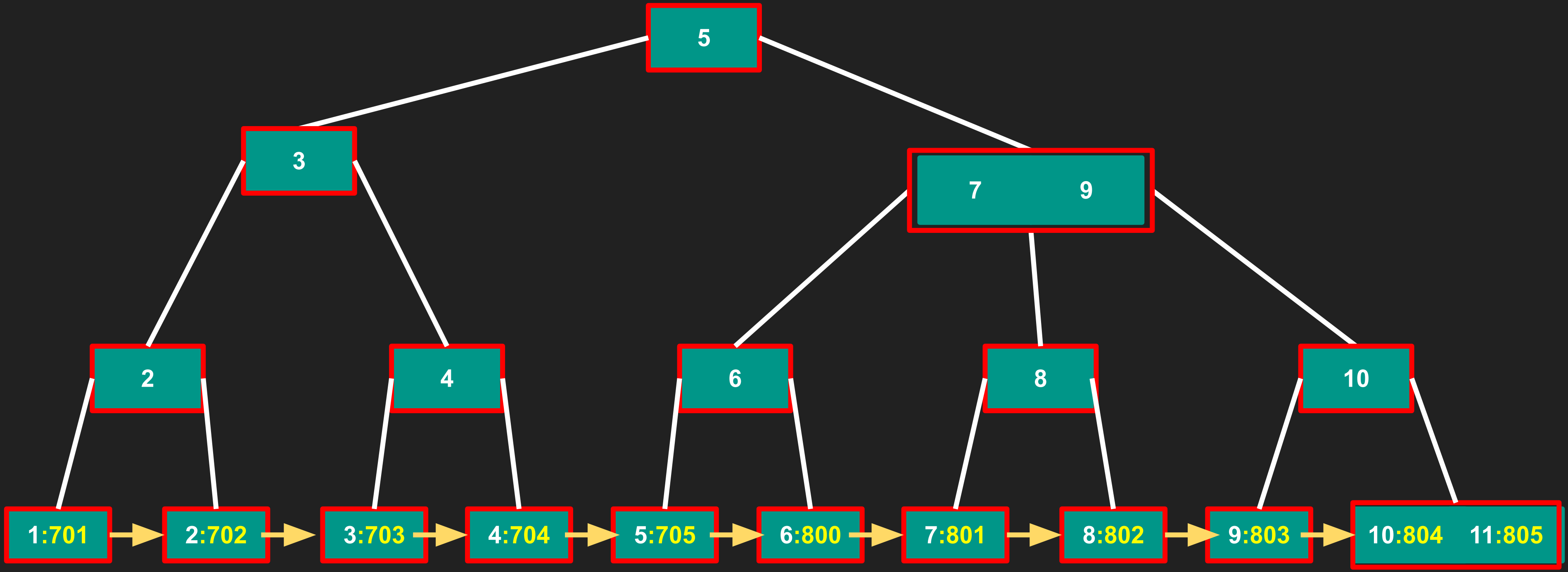


TID	ID	,	Name
701	1	,	John
702	2	,	Ali
703	3	,	Rick
704	4	,	Sara
705	5	,	Edmond
800	6	,	Ion
801	7	,	Edmond
802	8	,	Pete
803	9	,	Oliver
804	10	,	Yong
805	11	,	Xui

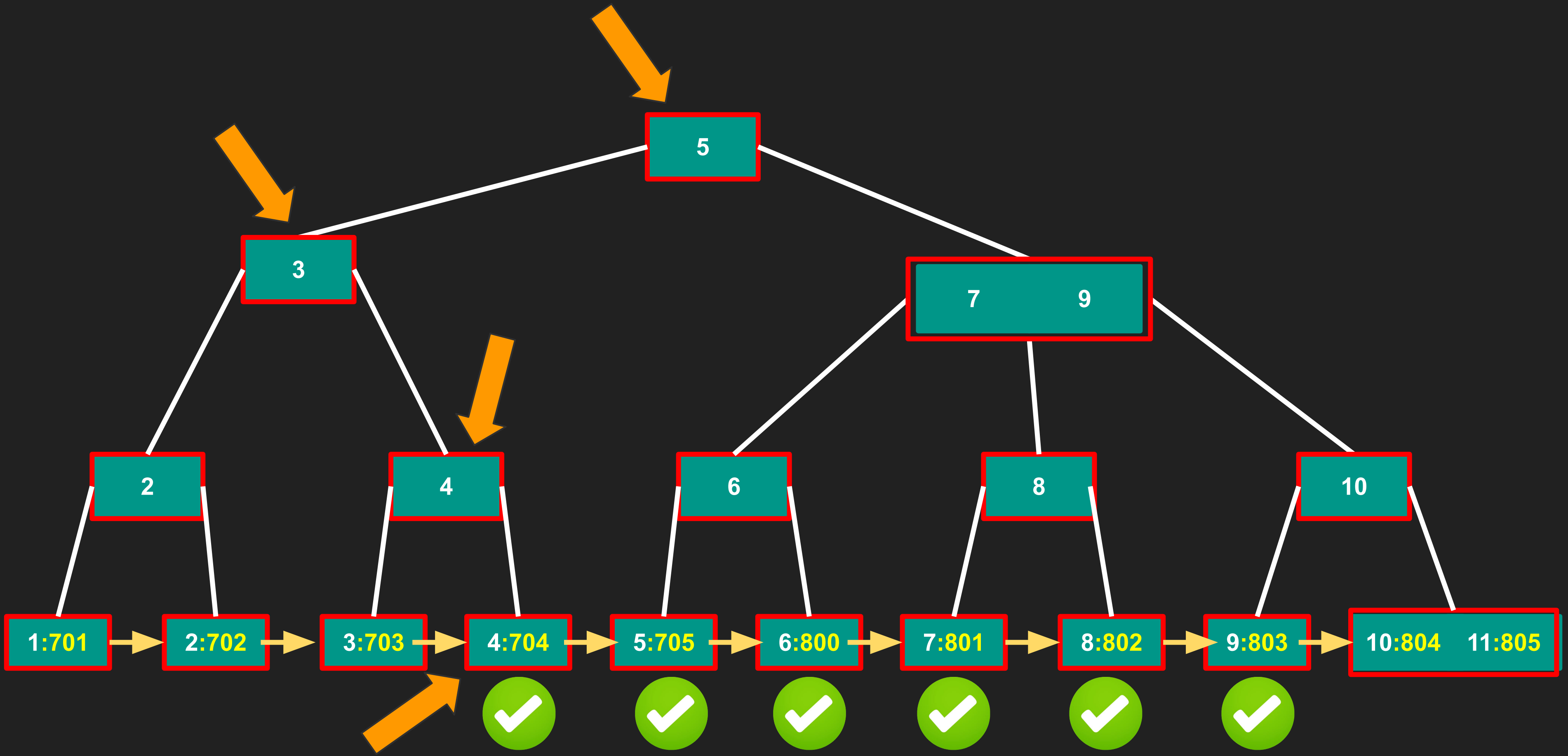
B+Tree

- Exactly like B-Tree but only stores keys in internal nodes
- Values are only stored in leaf nodes
- Internal nodes are smaller since they only store keys and they can fit more elements
- Leaf nodes are “linked” so once you find a key you can find all values before and after that key.
- Great for range queries

B+Tree of Degree 3



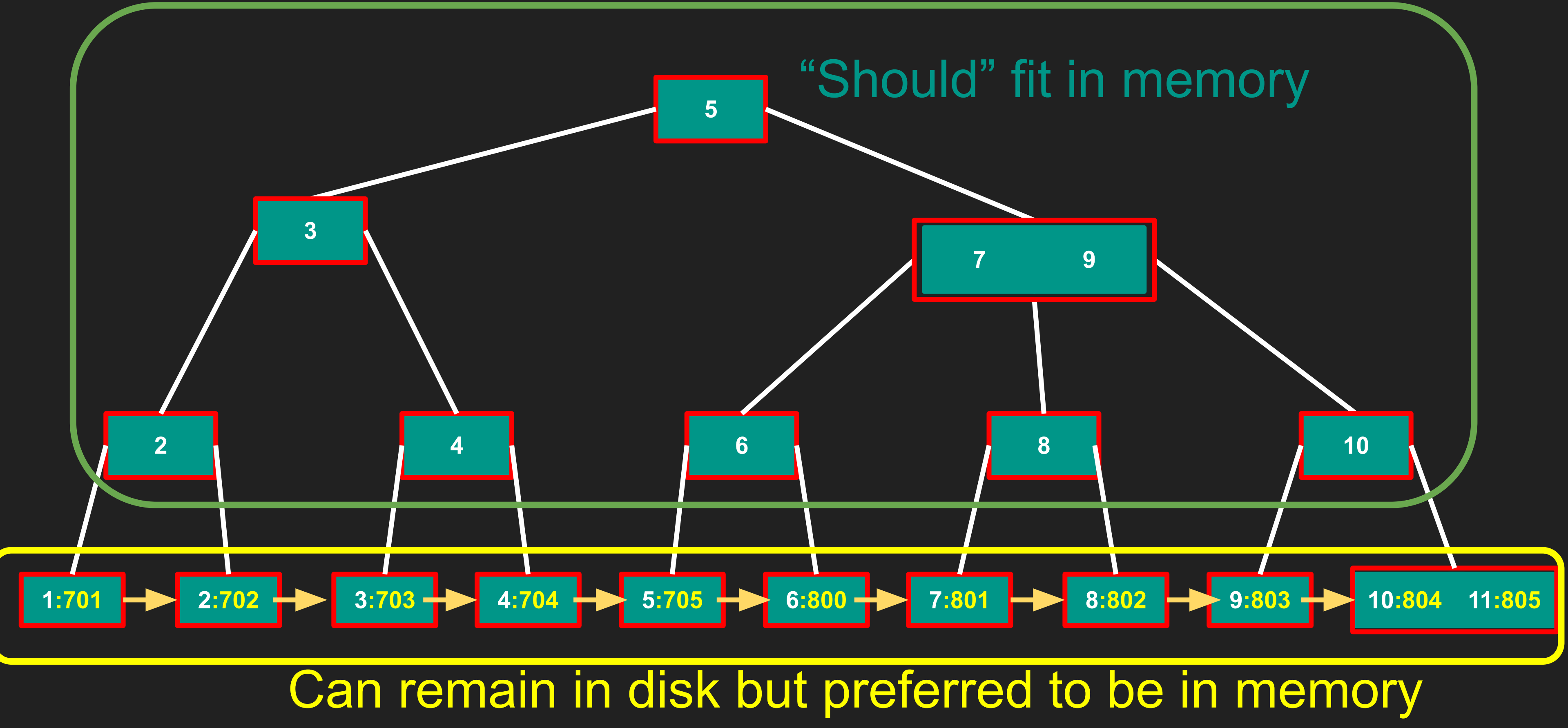
Find rows (ID between (4 and 9)



B+Tree & DBMS Considerations

- Cost of leaf pointer (cheap)
- 1 Node fits a DBMS page (most DBMS)
- Can fit internal nodes easily in memory for fast traversal
- Leaf nodes can live in data files in the heap
- Most DBMS systems use B+Tree

B+Tree Storage cost



Storage Cost in Postgres vs MySQL

- B+Trees secondary index values can either point directly to the tuple (Postgres) or to the primary key (MySQL)
- If the Primary key data type is expensive this can cause bloat in all secondary indexes for databases such MySQL (innoDB)
- Leaf nodes in MySQL (InnoDB) contains the full row since its an IOT / clustered index

Summary

- Full Table Scans
- B-Tree
- B-Tree limitations
- B+Tree
- B+Tree Considerations
- B+Tree storage cost in MySQL vs Postgres