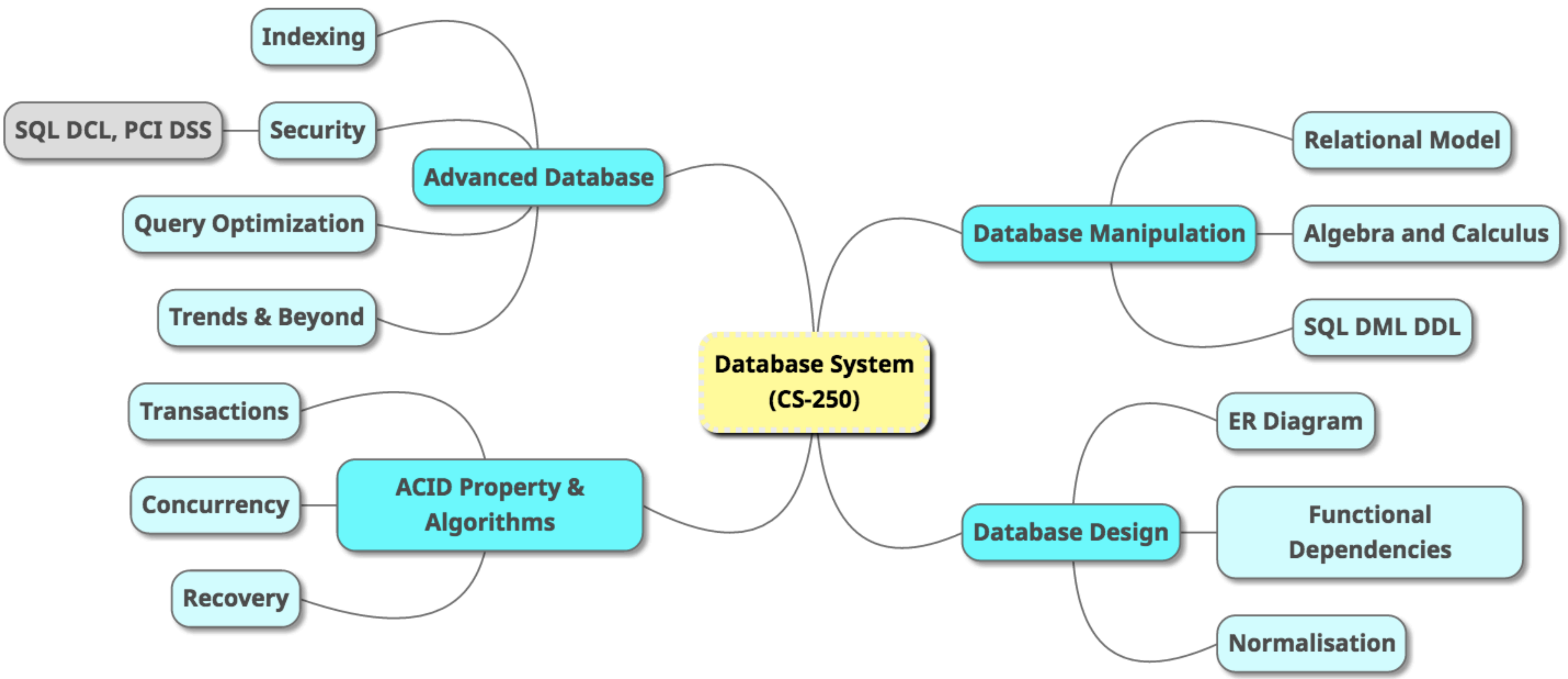
Overview



Indexing
SQL DCL, PCI DSS
Security
Advanced Database
Query Optimization
Trends & Beyond
Relational Model
Database Manipulation
Algebra and Calculus
SQL DML DDL
Database System (CS-250)
Transactions
ACID Property & Algorithms
Concurrency
Recovery
ER Diagram
Database Design
Functional Dependencies
Normalisation

B-Trees

Gary KL Tam

Department of Computer Science
Swansea University

Overview

Organisation of files is important, especially for gigantic dataset. Not only for fast access to the data, but also to achieve the following goals:

- Efficient use of storage space.
- Minimising the need for reorganisation.
- Accommodating growth.

All commercial database systems provide indexing mechanisms to accelerate the processing of SQL queries. In this lecture, we will discuss a variant of the B-tree, which is the one of the most important index in relational databases.

Data storage

Speed/Price comparison (2012)

HDD, SATA2	SSD, SATA3	RAM, DDR5
		
60-150MB/s	250-600MB/s	34000-52000 MB/s.
£/MB	£££/MB	£££££/MB

Data storage

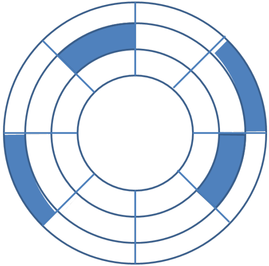
Speed/Price comparison (2012)

HDD, SATA2	SSD, SATA3	RAM, DDR5
		
60-150MB/s	250-600MB/s	34000-52000 MB/s.
£/MB	£££/MB	£££££/MB

Many DBMS is 20+ years old (e.g. Bank). Those are a lot slower!

Hard drive / Secondary Storage

- The tuples of a relation are stored in the hard disk.
- The hard disk is formatted into blocks, each of which has a fixed number of bytes, e.g., 4096.
- Each block has an address so that the database can retrieve any block by its address.
- Each tuple has a fixed number of bytes. Therefore, each block can store a fixed number of tuples.



Example

My USB stick:

```
D:\>chkdsk G:
The type of the file system is FAT32.
Volume NO NAME created 01/12/2013 16:41
Volume Serial Number is F086-8651
Windows is verifying files and folders...
File and folder verification is complete.
Windows has checked the file system and found no problems.
    7,800,304 KB total disk space.
          268 KB in 11 hidden files.
        9,280 KB in 2,233 folders.
    1,947,068 KB in 13,937 files.
    5,843,684 KB are available.

        4,096 bytes in each allocation unit.
    1,950,076 total allocation units on disk.
    1,460,921 allocation units available on disk.
```

Example

PROF(pid, name, rank, salary)

- pid and salary are of type integer - each 4 bytes
- name is of type char(20) - 20 bytes
- rank is of type char(8) - 8 bytes

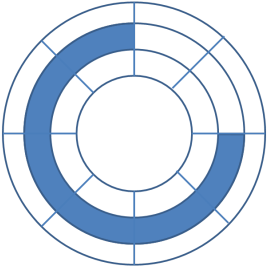
- Then, a tuple occupies 4 + 20 + 8 + 4 = 36 bytes.
- Hence, a block of 4096 bytes can store $\lfloor 4096/36 \rfloor = 113$ tuples.
- Therefore, if PROF has 1,000,000,000 tuples, then the table occupy $\lceil 1000000/113 \rceil = 8{,}849{,}558$ blocks.

$\lfloor\rfloor$ is the floor function. e.g. $\lfloor 12.7 \rfloor = 12$
$\lceil\rceil$ is the ceiling function. e.g. $\lceil 12.2 \rceil = 13$

SQL Query

select * from PROF where pid = 61

- By default, the database answers the above query by scanning the entire PROF table.
- Namely, it needs to read all the 8,849,558 blocks
- Assume sequential access: 60MB/s, 4096 Bytes/block

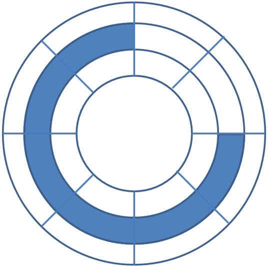


how long it takes?

SQL Query

select * from PROF where pid = 61

- By default, the database answers the above query by scanning the entire PROF table.
- Namely, it needs to read all the 8,849,558 blocks
- Assume sequential access: 60MB/s, 4096 Bytes/block



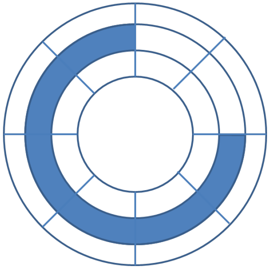
how long it takes?

- 8,849,558 blocks $\times$ 4096 bytes/block = 33.7GB

SQL Query

select * from PROF where pid = 61

- By default, the database answers the above query by scanning the entire PROF table.
- Namely, it needs to read all the 8,849,558 blocks
- Assume sequential access: 60MB/s, 4096 Bytes/block



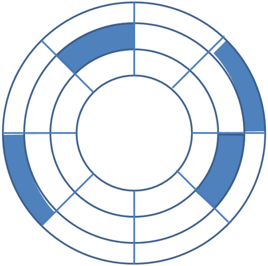
how long it takes?

- 8,849,558 blocks $\times$ 4096 bytes/block = 33.7GB
- 33.7GB / 60MB/s $\approx$ 10 mins. average 5 mins.

SQL Query

select * from PROF where pid = 61

- By default, the database answers the above query by scanning the entire PROF table.
- Namely, it needs to read all the 8,849,558 blocks
- Assume random access, and each disk (block) access takes 0.01 second:

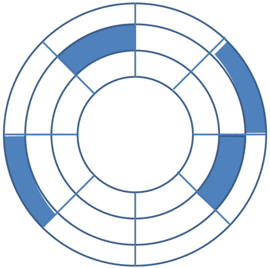


how long it takes?

SQL Query

select * from PROF where pid = 61

- By default, the database answers the above query by scanning the entire PROF table.
- Namely, it needs to read all the 8,849,558 blocks
- Assume random access, and each disk (block) access takes 0.01 second:



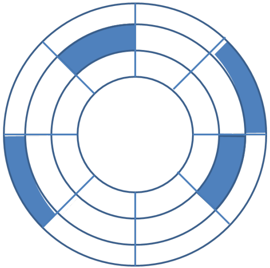
how long it takes?

- 8,849,558 $\times$ 0.01 seconds = 24.5 hrs. average 12 hrs.

SQL Query

select * from PROF where pid = 61

- By default, the database answers the above query by scanning the entire PROF table.
- Namely, it needs to read all the 8,849,558 blocks
- Assume random access, and each disk (block) access takes 0.01 second:



how long it takes?

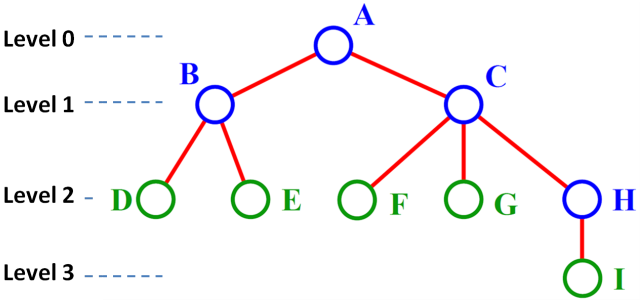
- 8,849,558 $\times$ 0.01 seconds = 24.5 hrs. average 12 hrs.
- An B-tree index reduce the query cost to merely 5 blocks, i.e. 0.05s!

Trees

Trees

- Tree a special data structure in computer science.
- This is NOT a programming course.
- But you need to understand the concept - how B-tree works.

Teminology

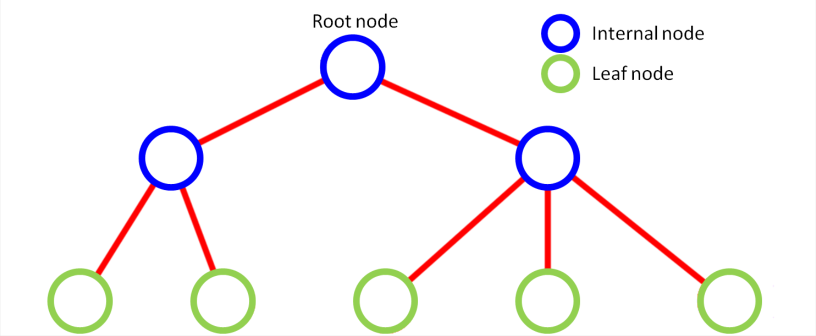


- A is the **root** node.
- B is the **parent** of D and E; D and E are the **children** of B.
- D, E, F, G, I are **external** nodes, or **leaves**.
- A, B, C, H are **internal** nodes.
- The **depth (level)** of E is 2.
- The **height** of the tree is 3 - maximum level.

B-Tree is a Tree

What is special about B-Tree?

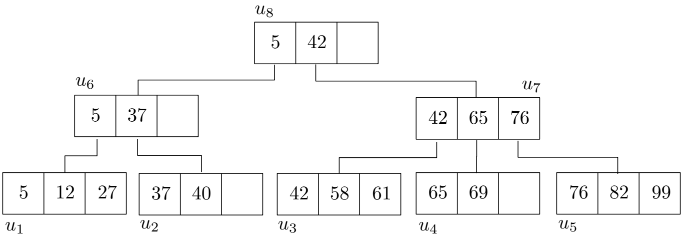
- It is balanced, i.e. all leaf nodes are at the same level.



Let S be a set of elements that is comparable, and to be stored. A B-tree of parameter B is a tree where:

- It is balanced
- Elements are stored in the leaf nodes.
- Each leaf node contains consecutive elements in S.
- Every leaf node has between $\lceil \frac{B}{2} \rceil$ and B elements, unless it is the only node in the tree
- Every internal node has between $\lceil \frac{B}{2} \rceil$ and B child nodes unless it is the root.
- The root has between 2 and B child nodes
- If node p is the parent of node u, then p stores a routing element that equals the smallest element stored in the leaf nodes in the subtree u.

An example with $B = 3$:



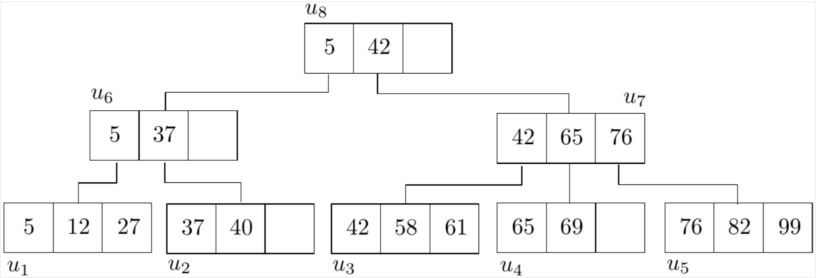
- Leaf nodes: $u_1, \ldots, u_5$.
- Internal nodes: $u_6, \ldots, u_8$.
- As an example of routing element, consider 5 in node u_8. This is the smallest element in the leaf nodes (u_1 and u_2) that are in the subtree of u_6.

Given an element q, we can efficiently determine whether $q \in S$. Furthermore, if the answer is yes, we can also find the leaf element corresponding to q.

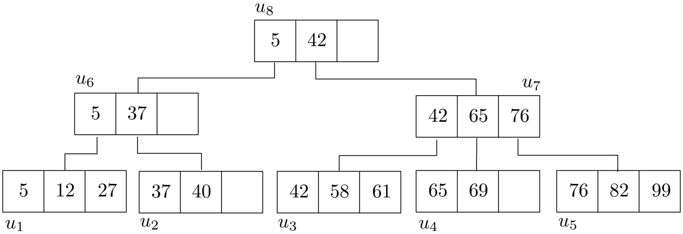
algorithm FindElement(q)

```
u ← the root of the B-tree
while u is an internal node
    e ← the predecessor of q among the routing elements in u
    if e does not exist
        return FALSE /* q does not exist */
    else
        u ← the child node of e
if q exists in u
    return TRUE
return FALSE
```

Are 61 and 38 in the tree?



$e \leftarrow$ the predecessor of q among the routing elements in node u
e : the largest routing element $\leq q$



- Given $q = 61$, we find it by accessing u_8, u_7 and u_3.
- Given $q = 38$, we declare its absence by accessing u_8, u_6 and u_2.

A B-tree is a dynamic structure, namely, it can be updated whenever an element is inserted or deleted in S.

Insertion

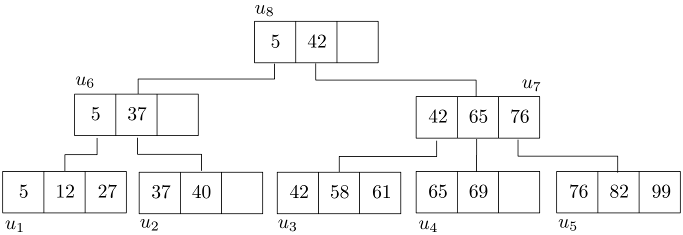
To insert an element e:

1. Find the leaf node u where e should be inserted.

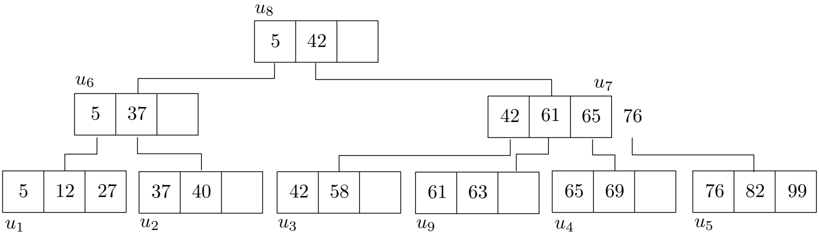
Think

How?

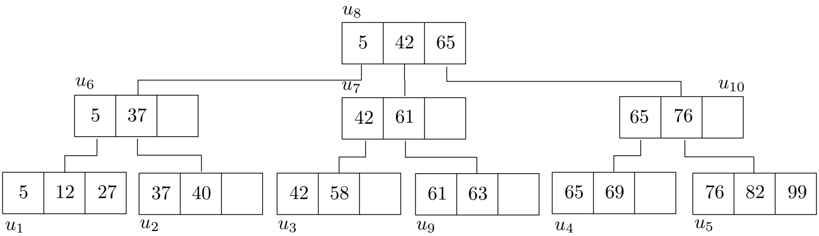
2. Add e to u.
3. If u overflows (i.e., having more than B elements), split it into two nodes u_1 and u_2 by distributing the elements evenly.
    - If u is the root, create a new root with u_1 and u_2 as the child nodes.
    - Otherwise, let p be the parent of u. Make u_1 and u_2 the child nodes of p.



Example. Consider the insertion of element 63. It should be inserted into u_3. After adding 63 to u_3, the node overflows, and hence, is split. See the next slide.



Example (cont.). Now u_7 has 4 child nodes, thus overflows, and is split. See the next slide.

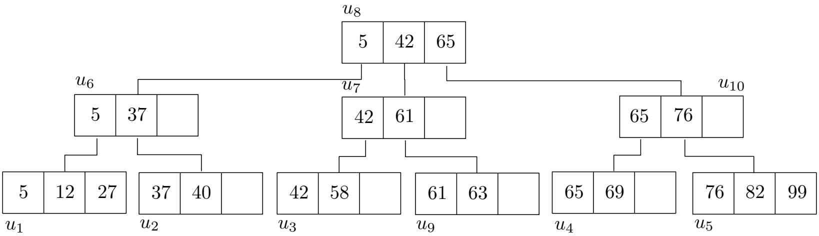


Example (cont.). Final situation.

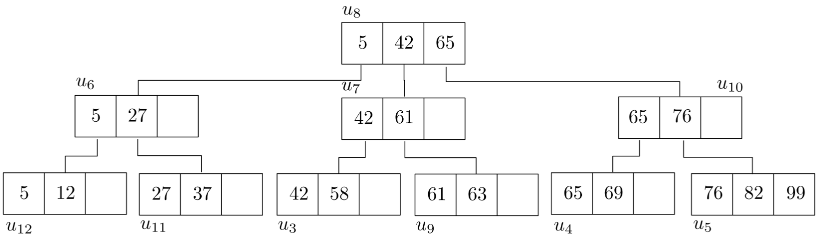
Deletion

To delete an element e:

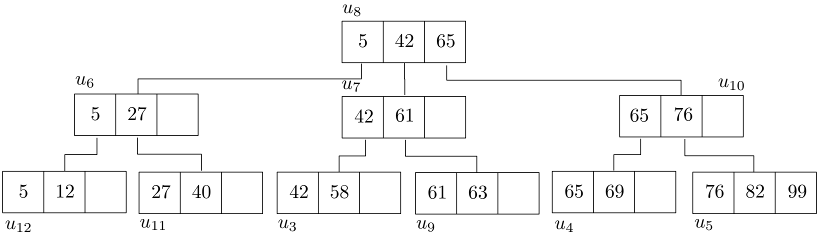
1. Find the leaf node u where e is stored.
2. Remove e from u.
3. If u underflows (i.e., having less than $B/2$ elements), merge it with a neighboring sibling (i.e., a node with the same parent p of u).
   - If the merged node has more than B elements, split.
4. Adjust the child nodes of p accordingly.
5. If p is the root and has only 1 child node left, remove p.
6. If p is not the root but underflows, handle it in the same fashion.



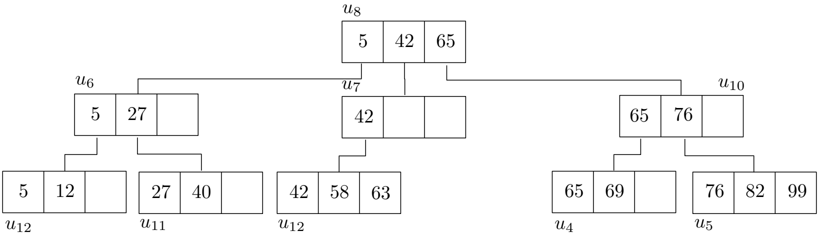
Example. Consider the deletion of element 40. Node u_2 underflows after 40 is gone. Hence, we merge u_1 with u_2. However, the resulting node has 4 elements, and therefore, needs to split. See the next slide.



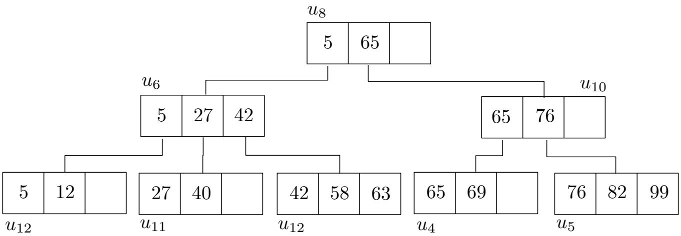
Example (cont.). Final situation.



Example. Consider the deletion of element 61. Node u_9 underflows after 61 is gone. Hence, we merge u_9 with u_2. See the next slide.



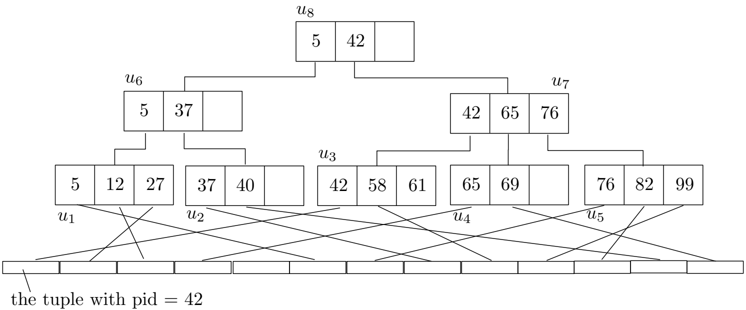
Example (cont.). Node u_7 underflows. It has two siblings u_6 and u_{10}, both of which can be merged with u_7. Suppose that we choose (arbitrarily) u_6 (you can figure out what happens with the other choice).



Example (cont.). Final situation.

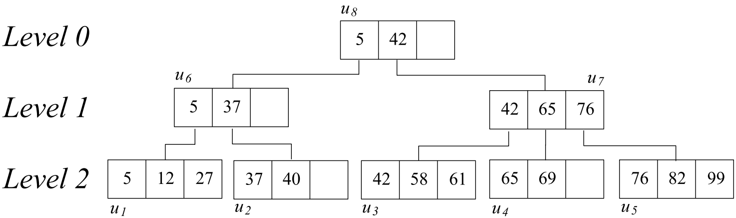
B-Tree on a Relation

The following figure shows how a B-tree built on the pid attribute of table PROF looks:



- Each node is stored in a block (i.e., B depends on the block size).
- Each pointer (a.k.a. link) is a block address.
- Each element stores a pointer to the block where the corresponding tuple resides.

Height of a B-tree



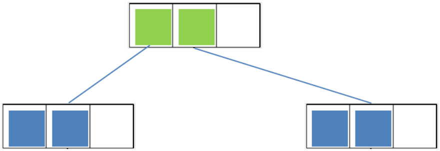
- All elements stored only in leaf nodes.
- Root at 0 level.
- Height of a B-tree: the maximum level

- What is the maximum height of a tree, given $|S|$ number of elements?

- What is the maximum height of a tree, given $|S|$ number of elements?
- Ideas: each node store the minimum #items; i.e., half full!

- What is the maximum height of a tree, given $|S|$ number of elements?
- Ideas: each node store the minimum #items; i.e., half full!

S = 4 x　　　B = 3



Root: min 2 items

Leaf node: min #items

$$\left\lceil \frac{B}{2} \right\rceil = \left\lceil \frac{3}{2} \right\rceil = 2$$

Total #items:

$$2 \times \left\lceil \frac{B}{2} \right\rceil = 4$$

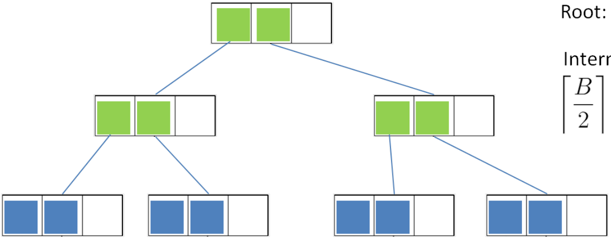
routing element

element in S, store at leaf nodes

Max height = 1

- What is the maximum height of a tree, given $|S|$ number of elements?

S = 8 x   B = 3



Root: min 2 items

Internal node: min #items

$$\left\lceil \frac{B}{2} \right\rceil = \left\lceil \frac{3}{2} \right\rceil = 2$$

Leaf node: min #items

$$\left\lceil \frac{B}{2} \right\rceil = \left\lceil \frac{3}{2} \right\rceil = 2$$

Total #items:

$$2 \times \left\lceil \frac{B}{2} \right\rceil \times \left\lceil \frac{B}{2} \right\rceil = 8$$

Max height = 2

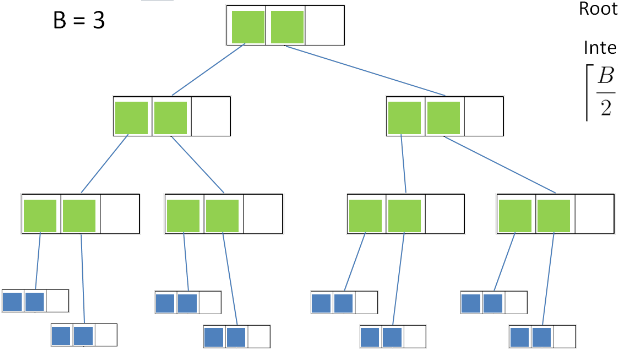
routing element

element in S, store at leaf nodes

- What is the maximum height of a tree, given $|S|$ number of elements?

S = 16 x

B = 3



Root: min 2 items

Internal node: min #items

$$\left\lceil \frac{B}{2} \right\rceil = \left\lceil \frac{3}{2} \right\rceil = 2$$

Leaf node: min #items

$$\left\lceil \frac{B}{2} \right\rceil = \left\lceil \frac{3}{2} \right\rceil = 2$$

Total #items:

$$2 \left\lceil \frac{B}{2} \right\rceil^{h} = 16$$

Max height (h) = 3

routing element

element in S, store at leaf nodes

Height and size of a B-tree

The relation of height and size of a B-tree:

$$n = 2\left\lceil \frac{B}{2} \right\rceil^h$$

Given $n = |S|$ number of elements, the max height of a B-tree is:

$$2\left\lceil \frac{B}{2} \right\rceil^h = n$$

$$\left\lceil \frac{B}{2} \right\rceil^h = \frac{n}{2}$$

$$\log_{\lceil \frac{B}{2} \rceil} \left\lceil \frac{B}{2} \right\rceil^h = \log_{\lceil \frac{B}{2} \rceil} \frac{n}{2}$$

$$h = \log_{\lceil \frac{B}{2} \rceil} \frac{n}{2}$$

Automatic Number Plate Recognition - Police

Imagine you creating a database (about 22 million cars) that allows the Police to instantly look up any car in the UK using the number plate. Assuming each disk read takes 0.01 seconds, and 100 cars can be stored on a block.

Question

How many disk accesses and time is needed (in the worst case):

- use a sequential search?
- use a B-Tree of order $B = 100$?

ASSOCIATION OF
CHIEF POLICE OFFICERS

Example

Sequential Search:

- In the worst case we will need to make 22,000,000/100=220,000 disk accesses:
- 220,000 $\times$ 0.01 seconds = 36 minutes 40 seconds.
- On average 18m20s.

Example

B-Tree Search:

- B-Tree of order $B = 100$, $n = 22,000,000$
- Use formula: $h = \log_{\lceil \frac{B}{2} \rceil} \frac{n}{2}$
- $h = \log_{50} 11,000,000$
- $h = 5$
- At most 5 disk accesses (including 1 for the root node at level 0) in order to find any record in our tree of 22 million records giving a time of 5×0.01 s $= 0.05$ seconds.

B-Tree on a Relation

Think: How would you answer the following query using a B-tree?

select * from PROF where pid $= 61$

PROF Example

SQL Query

select * from PROF where pid = 61

- Given PROF has 1,000,000,000 tuples.
- B-tree of order $B = 113$

PROF Example

SQL Query

select * from PROF where pid = 61

- Given PROF has 1,000,000,000 tuples.
- B-tree of order $B = 113$
- Use formula: $h = \log_{\lceil \frac{B}{2} \rceil} \frac{n}{2}$
- $h = \log_{57} 500,000,000$
- $h = 5$

PROF Example

SQL Query

select * from PROF where pid = 61

- Given PROF has 1,000,000,000 tuples.
- B-tree of order $B = 113$
- Use formula: $h = \log_{\lceil \frac{B}{2} \rceil} \frac{n}{2}$
- $h = \log_{57} 500,000,000$
- $h = 5$

Conclusion

B-Trees with high branching factors B are far more efficient when data needs to be accessed on slow external memory devices.

Statement for Index Creation

Most RDBMS (e.g. MySQL, MSSQL, Oracle):

- index is automatically built for each **primary key**.
- index is used to implement **unique** keyword.

create index ⟨index-name⟩ on ⟨table-name⟩ (⟨attribute-name⟩)

Example. create index prof_index on prof (sal)

Note, sal is a non-key attribute in this example.

Muddiest Points - Past student feedback

Why teach binary tree again? We have already learnt that in Year1!

What is 'B' really?

Reference:
https://www.quora.com/What-does-the-B-stand-for-in-B-Tree

Muddiest Points - Past student feedback

Why teach binary tree again? We have already learnt that in Year1!

- Binary Tree - at most two child nodes for each node. It may not balance.

What is ‘B’ really?

Reference:
`https://www.quora.com/What-does-the-B-stand-for-in-B-Tree`

Muddiest Points - Past student feedback

Why teach binary tree again? We have already learnt that in Year1!

- Binary Tree - at most two child nodes for each node. It may not balance.
- B-Tree - at least B/2 number of child nodes per each internal node. It is SELF-BALANCED.

What is 'B' really?

Reference:
https://www.quora.com/What-does-the-B-stand-for-in-B-Tree

Muddiest Points - Past student feedback

Why teach binary tree again? We have already learnt that in Year1!

- Binary Tree - at most two child nodes for each node. It may not balance.
- B-Tree - at least B/2 number of child nodes per each internal node. It is SELF-BALANCED.

What is 'B' really?

- No one knows what 'B' stands for as the authors have not mentioned it.

Reference:
`https://www.quora.com/What-does-the-B-stand-for-in-B-Tree`

Muddiest Points - Past student feedback

Why teach binary tree again? We have already learnt that in Year1!

- Binary Tree - at most two child nodes for each node. It may not balance.
- B-Tree - at least B/2 number of child nodes per each internal node. It is SELF-BALANCED.

What is 'B' really?

- No one knows what 'B' stands for as the authors have not mentioned it.
- It would mean "Balance", "Bayer" (one of the authors), "Boeing" company...

Reference:
`https://www.quora.com/What-does-the-B-stand-for-in-B-Tree`

Muddiest Points - Past student feedback

Why teach binary tree again? We have already learnt that in Year1!

- Binary Tree - at most two child nodes for each node. It may not balance.
- B-Tree - at least B/2 number of child nodes per each internal node. It is SELF-BALANCED.

What is 'B' really?

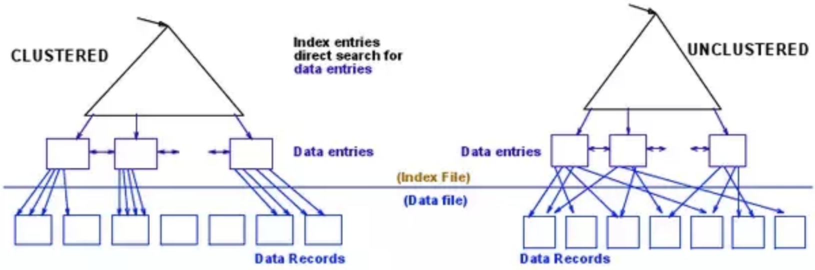
- No one knows what 'B' stands for as the authors have not mentioned it.
- It would mean "Balance", "Bayer" (one of the authors), "Boeing" company...
- It is definitely NOT "Binary".

Reference:
`https://www.quora.com/What-does-the-B-stand-for-in-B-Tree`

Clustered vs Non-Clustered Index

Clustered index (Out-Of-Syllabus / Out-Of-Exam)

- requires the data records be ***sorted*** on disk as well.
- higher update overhead, but quicker
- considered if necessary, but only for primary key
- ***different*** Big-O for different ***query types*** (see below)
- frequent interview question (e.g. will there speed benefit for ... situation?)



Extra reading (outside exam & syllabus):
https://www.quora.com/How-are-clustered-and-non-clustered-indexing-implemented-internally