Practicing B+ Trees

Database Applications - Recitation 10
Zeinab Khalifa
March 26th, 2020
Exercise (1)

Preparing our B+ Tree
Consider the relation

\[
\text{Student}(\text{sid: type, name: char, major: char, gpa: double})
\]

A B+ Tree Page

My index is on \text{sid}

Let's assume our mini disk has a block size of 64 bytes

How can we build our B+ tree?

We need to know how many keys (order)...

(\text{n+1}) Pointers
Let's assume our mini disk has a block size of 64 bytes.
To fit a B+ tree page into a disk block of size 64 bytes

Let's assume our mini disk has a block size of 64 bytes

A B+ Tree Page

K1  K2  ...  Kn

(n+1) Pointers
To fit a B+ tree page into a disk block of size 64 bytes

A B+ Tree Page

K1  K2  ...  K\text{\text{\text{n}}}

(n+1) Pointers
To fit a B+ tree page into a disk block of size 64 bytes

\[ \text{Size}(K_1, K_2, \ldots, K_n) \leq 64 \]

A B+ Tree Page

\((n+1)\) Pointers
To fit a B+ tree page into a disk block of size 64 bytes
Assume that: sid/key size = 4 bytes and pointers are of size 8 bytes

\[ \text{Size}(\text{A B+ Tree Page}) \leq 64 \]
To fit a B+ tree page into a disk block of size 64 bytes
Assume that: sid/key size = 4 bytes and pointers are of size 8 bytes

\[
\text{Size}((K_1, K_2, \ldots, K_n, (n+1) \text{ Pointers})) \leq 64
\]

\[
4n + 8(n + 1) \leq 64
\]

Solving for \( n \)

\[
n \leq 5.3
\]

Maximum number of keys \( 2d = 5 \),
Tree order: \( d = 2 \)
Exercise (2)

Let’s start populating our data in the Student relation
<table>
<thead>
<tr>
<th>SID</th>
<th>Name</th>
<th>Major</th>
<th>GPA</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>James Smith</td>
<td>Computer Science</td>
<td>2.91</td>
</tr>
<tr>
<td>10</td>
<td>Michael Smith</td>
<td>Computer Science</td>
<td>3.22</td>
</tr>
<tr>
<td>15</td>
<td>Robert Smith</td>
<td>Biological Sciences</td>
<td>2.59</td>
</tr>
<tr>
<td>20</td>
<td>Maria Hernandez</td>
<td>Computer Science</td>
<td>3.00</td>
</tr>
<tr>
<td>25</td>
<td>Michael Garcia</td>
<td>Computational Biology</td>
<td>2.54</td>
</tr>
<tr>
<td>30</td>
<td>Maria Garcia</td>
<td>Information Systems</td>
<td>4.0</td>
</tr>
<tr>
<td>50</td>
<td>..</td>
<td>..</td>
<td>..</td>
</tr>
<tr>
<td>55</td>
<td>..</td>
<td>..</td>
<td>..</td>
</tr>
<tr>
<td>60</td>
<td>..</td>
<td>..</td>
<td>..</td>
</tr>
<tr>
<td>65</td>
<td>..</td>
<td>..</td>
<td>..</td>
</tr>
<tr>
<td>75</td>
<td>..</td>
<td>..</td>
<td>..</td>
</tr>
<tr>
<td>80</td>
<td>..</td>
<td>..</td>
<td>..</td>
</tr>
<tr>
<td>85</td>
<td>..</td>
<td>..</td>
<td>..</td>
</tr>
<tr>
<td>90</td>
<td>..</td>
<td>..</td>
<td>..</td>
</tr>
<tr>
<td>28</td>
<td>..</td>
<td>..</td>
<td>..</td>
</tr>
</tbody>
</table>

Insert into Student (SID, Name, Major, GPA) values
(5, "", "", ",")
(10,"","",""),
(15,"","",""),
(20,"","",""),
(25,"","",""),
(30,"","",""),
...
(90,"","",""),
(28,"","",""),
Now insert key 70

The leaf page is full but the index is not
Now insert key 70

The leaf page is full but the index is not
Now insert key 70
The leaf page is full but the index is not
Now insert key 70
The leaf page is full but the index is not
Now insert key 70

The leaf page is full but the index is not
Now insert key 70
The leaf page is full but the index is not
Now insert key 95

The leaf page and the index are full
Now insert key 95

The leaf page and the index are full
Now insert key 95

The leaf page and the index are full
Now insert key 95

The leaf page and the index are full
Now insert key 95
The leaf page and the index are full
Now insert key 95

The leaf page and the index are full
Now insert key 95

The leaf page and the index are full
Now insert key 95

The leaf page and the index are full.
Now insert key 95

The leaf page and the index are full

60 to rise up
Exercise (3)

Let’s play with numbers...
After inserting 6000 records, we are curious to know how high our tree has become!

 Alternative 3

\[ K^* : (k_1, [\text{ridx, ridy}]) \]
After inserting 6000 records, we are curious to know how high our tree has become!

- Key size = 4 bytes
- Pointer Size = 8 bytes
- Disk block = 64 bytes.
- $\text{Avg}(\text{Size}(\text{rid-list})) = 2$
- $d = 2$
- 6000 total records

<table>
<thead>
<tr>
<th>SID</th>
<th>Name</th>
<th>Major</th>
<th>GPA</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>James Smith</td>
<td>Computer Science</td>
<td>2.91</td>
</tr>
<tr>
<td>10</td>
<td>Michael Smith</td>
<td>Computer Science</td>
<td>3.22</td>
</tr>
<tr>
<td>15</td>
<td>Robert Smith</td>
<td>Biological Sciences</td>
<td>2.59</td>
</tr>
<tr>
<td>20</td>
<td>Maria Hernandez</td>
<td>Computer Science</td>
<td>3.00</td>
</tr>
<tr>
<td>25</td>
<td>Michael Garcia</td>
<td>Computational Biology</td>
<td>2.54</td>
</tr>
<tr>
<td>30</td>
<td>Maria Garcia</td>
<td>Information Systems</td>
<td>4.0</td>
</tr>
<tr>
<td>6000</td>
<td>Hammoud</td>
<td>Computer Science</td>
<td>4.0</td>
</tr>
</tbody>
</table>
After inserting 6000 records, we are curious to know how high our tree has become!

- Key size = 4 bytes
- Pointer Size = 8 bytes
- Disk block = 64 bytes.
- Avg(\text{Size(rid-list)}) = 2
- d = 2
- 6000 total records
After inserting 6000 records, we are curious to know how high our tree has become!

$log_{\#\text{pointers}}(\#\text{leaves}) + 1$

- Key size = 4 bytes
- Pointer Size = 8 bytes
- Disk block = 64 bytes.
- $\text{Avg}(\text{Size(rid-list)}) = 2$
- $d = 2$
- 6000 total records
After inserting 6000 records, we are curious to know how high our tree has become!

\[ \log_5(\text{#leaves}) + 1 \]

- Key size = 4 bytes
- Pointer Size = 8 bytes
- Disk block = 64 bytes.
- Avg(Size(rid-list)) = 2
- \( d = 2 \)
- 6000 total records
After inserting 6000 records, we are curious to know how high our tree has become!

\[ \log_5(\#leaves) + 1 \]

Key size = 4 bytes
- Pointer Size = 8 bytes
- Disk block = 64 bytes.
- \( \text{Avg}(\text{Size(rid-list)}) = 2 \)
- \( d = 2 \)
- 6000 total records
After inserting 6000 records, we are curious to know how high our tree has become!

\[ \log_5(\#leaves) + 1 \]

- Key size = 4 bytes
- Pointer Size = 8 bytes
- Disk block = 64 bytes.
- \( \text{Avg}(\text{Size}(\text{rid-list})) = 2 \)
- \( d = 2 \)
- 6000 total records

\( K^* : (k1, [\text{ridx, ridy}]) \)

Size of \( K^* = 20 \) BYTES
After inserting 6000 records, we are curious to know how high our tree has become!

- Key size = 4 bytes
- Pointer Size = 8 bytes
- Disk block = 64 bytes.
- \( \text{Avg}(\text{Size}(\text{rid-list})) = 2 \)
- \( d = 2 \)
- 6000 total records

Size of \( K^* \) = 20 BYTES

How many can we fit in 1 disk block/page?

\[ \log_5(\#\text{leaves}) + 1 \]
After inserting 6000 records, we are curious to know how high our tree has become!

- Key size = 4 bytes
- Pointer Size = 8 bytes
- Disk block = 64 bytes.
- \( \text{Avg(\text{Size(rid-list)})} = 2 \)
- \( d = 2 \)
- 6000 total records

\[ \log_5(\#\text{leaves}) + 1 \]

Size of \( K^* \) = 20 BYTES

How many can we fit in 1 disk block/page?

\[ \frac{64}{20} \approx 3 \, K^* \]
After inserting 6000 records, we are curious to know how high our tree has become!

- Key size = 4 bytes
- Pointer Size = 8 bytes
- Disk block = 64 bytes
- Avg(\text{Size(rid-list)}) = 2
- \( d = 2 \)
- 6000 total records

Each K* has 2 Records
Total = 3 * 2 = 6 Records in a leaf

\[ \text{Size of K}^* = 20 \text{ BYTES} \]

\[ \log_5(\text{#leaves}) + 1 \]

How many can we fit in 1 disk block/page?

\[ \frac{64}{20} \approx 3 \text{ K}^* \]
After inserting 6000 records, we are curious to know how high our tree has become!

- Key size = 4 bytes
- Pointer Size = 8 bytes
- Disk block = 64 bytes.
- \( \text{Avg(Size(rid-list))} = 2 \)
- \( d = 2 \)
- 6000 total records

\[ \log_5(\#\text{leaves}) + 1 \]

Each \( K^* \) has 2 Records
Total = 3 * 2 = 6 Records in a leaf

Size of \( K^* \) = 20 BYTES

How many can we fit in 1 disk block/page?
\[ \frac{64}{20} \approx 3 \text{ } K^* \]
After inserting 6000 records, we are curious to know how high our tree has become!

$$\log_5(\#\text{leaves}) + 1$$

- Key size = 4 bytes
- Pointer Size = 8 bytes
- Disk block = 64 bytes.
- $\text{Avg}(\text{Size(rid-list)}) = 2$
- $d = 2$
- 6000 total records
After inserting 6000 records, we are curious to know how high our tree has become!

\[
\text{log}_5 (1000) + 1 = 5.2 \approx 6 \text{ levels}
\]

- Key size = 4 bytes
- Pointer Size = 8 bytes
- Disk block = 64 bytes.
- \( \text{Avg}(\text{Size(rid-list)}) = 2 \)
- \( d = 2 \)
- 6000 total records
Exercise (4)
Happy students! 😊
Happy students! 😊

We decided to increase the GPA of each student by 0.5 for all students with GPA < 4.00. Accordingly, we wrote this query..

```
UPDATE Students SET GPA=GPA+0.5 WHERE GPA < 4.00
```
Happy students! 😊

We decided to increase the GPA of each student by 0.5 for all students with GPA < 4.00. Accordingly, we wrote this query.

```
UPDATE Students SET GPA=GPA+0.5 WHERE GPA < 4.00
```

**Oopps!**

After running this query, we found that all students ended up with a GPA 4.00.

Why do you think this happened?

What are some possible solutions?