Waves of misery after index creation

Nikolaus Glombiewski\textsuperscript{1}, Bernhard Seeger\textsuperscript{1}, Goetz Graefe\textsuperscript{2}

\textsuperscript{1}University of Marburg
\textsuperscript{2}Google Inc.
Outline

• Problem Assessment
• Basic Solution
• Ideal Solution
• Practical Remedies
• Experimental Evaluation
• Conclusion
Outline

• Problem Assessment
  • Basic Solution
  • Ideal Solution
• Practical Remedies
• Experimental Evaluation
• Conclusion
Indexes: Pros & Cons

• Pros
  • Fast lookups
  • Fast ordered range scans

➔ Best supported by bulk loading a perfect secondary b-tree

• Cons
  • Maintenance cost
  • Robustness of performance over time
Creation of a Perfect B-tree

1, 5, 8, 15, 17, 19, 41, 50, 90, 100, 120, 142

Building B-Tree
Max Nodesize = 3

15, 41, 100

1, 5, 8
15, 17, 19
41, 50, 90
100, 120, 142
Subsequent Insertions on a Perfect B-tree

Max Nodesize = 3

Insert Batch

4, 10, 22, 60, 102, 150

15, 41, 100

1, 5, 8
15, 17, 19
41, 50, 90
100, 120, 142
Subsequent Insertions on a Perfect B-tree

Max Nodesize = 3

Insert Batch

15, 41, 100

1, 4, 5, 8, 10

Node Split

14, 17, 19, 22

Node Split

41, 50, 60, 90

Node Split

100, 102, 120, 142, 150

Node Split

4, 10, 22, 60, 102, 150

=> Immediate, widespread node splits after index creation
Problem of Subsequent Insertions

• Splits of almost all leaves within a short time period
  • high I/O load
  • low buffer utilization
  • low query performance due to contention

• Status quo database solution: Leave free space (e.g. 30%)
  • Oracle, SQL Server, DB2, ...
Creation of a Perfect B-tree with free space

1, 5, 8, 15, 17, 19, 41, 50, 90, 100, 120, 142

Building B-Tree
Max Nodesize = 3, Utilization = 70%

8, 17, 41, 90, 120

1, 5
8, 15
17, 19
41, 50
90, 100
120, 142
Subsequent insertions

Max Nodesize = 3

Insert Batch

4, 10, 22, 60, 102, 150

8, 17, 41, 90, 120
Subsequent insertions

Max Node Size = 3

Insert Batch

4, 10, 22, 60, 102, 150

8, 17, 41, 90, 120

1, 4, 5
8, 10, 14
17, 19, 22
41, 50, 60
90, 100, 102
120, 142, 150
Continuation of insertions

Max Nodesize = 3

1, 4, 5 8, 10, 14 17, 19, 22 41, 50, 60 90, 100, 102 120, 142, 150

Insert Batch
6, 13, 38, 55, 95, 136
Continuation of insertions

Max Nodesize = 3

=> Delayed, widespread node splits after index creation
Limitations of the status quo

• The problem of splits is merely delayed
• Moreover, the problem occurs in waves
Problem Assessment – When does it occur?

- Loading Distribution = Insert Distribution
  - E.g.: Hash-Keys
Outline

• Problem Assessment
• Basic Solution
• Ideal Solution
• Practical Remedies
• Experimental Evaluation
• Conclusion
Basic Idea

• Do not leave constant free space while loading
Basic Idea – Insert Batch

=> *Distributing* node splits over time
Outline

• Problem Assessment
• Basic Solution
• Ideal Solution
• Practical Remedies
• Experimental Evaluation
• Conclusion
Ideal solution for predictable splits

Moving hills into valleys

Leaf Splits

Insert Batch (Batchsize 10,000)
Ideal solution (Leaf Nodes)

• $q_B = \text{Probability of a split after insertion}$
Ideal solution (Leaf Nodes)

• Fringe Analysis:
  \[
  \begin{pmatrix}
    q_B \\
    \vdots \\
    q_B
  \end{pmatrix}^2 = \tilde{q}(n)
  \]

• Insert-Operation:
  \[
  \tilde{q}(n) \ast \left( I + \frac{1}{n+1} T \right) = \tilde{q}(n + 1)
  \]

• Goal: \( \tilde{q}(n) = \tilde{q}(n + k) \) => A stable state
Ideal solution (Leaf Nodes)

• Goal: $\tilde{q}(n) = \tilde{q}(n + k) \implies$ A stable state

• Analyze Transition: $T \ast \begin{pmatrix} q_B \\ \vdots \\ q_B \end{pmatrix} = \vec{0}$

• Formula holds for $q_j = 1/(j+1)$
Ideal solution (Leaf Nodes)

• Intuition:
  • Few full pages split *immediately*
  • Many half full pages *eventually*

• Ideal solution 😊
  • ...for expected B-tree utilization
  • ...i.e., for Utilization of $\ln(2) = 69\%$ 😞
Outline

• Problem Assessment
• Basic Solution
• Ideal Solution
• Practical Remedies
• Experimental Evaluation
• Conclusion
Practical Remedies – Random

• While loading: Randomly pick around target utilization

nextPage() ➔ 80%+
x

80% ➔ 80%-
y
Practical Remedies – Suffix Truncation

• While loading: Search for shortest key within range

• Added compression effect
Outline

• Problem Assessment
• Basic Solution
• Ideal Solution
• Practical Remedies
• Experimental Evaluation
• Conclusion
Experimental Evaluation – Setup

• Procedure:
  • Records: 21 integers (84 bytes), normal distribution
  • Loading b-tree with 100,000 pages of 8KB
  • Inserting batches of 10,000 records

• Workstation:
  • AMD Ryzen7 2700X
  • 16GB memory
  • Java indexing library XXL
Experimental Evaluation – Random

![Graph showing leaf splits over insert batch size]

- Random 20% Range
- Random 10% Range
- Random 5% Range
- Constant 80%
Experimental Evaluation – Buffer Utilization
Outline

• Problem Assessment
• Basic Solution
• Ideal Solution
• Practical Remedies
• Experimental Evaluation
• Conclusion
Waves of Misery after index creation

- Loading secondary b-tree index in...
  - Write-intensive workloads
  - Loading distribution = Insert distribution
- Want to achieve predictable split performance:

![](https://cdn-images-1.medium.com/max/1600/1*ZjP65JlRufUc9GELlKyzWw.png)

Don't just leave *constant* free space in your tree nodes!

Work towards starting in the steady state of the b-tree.
Thank you for your attention!