

Designing an Index for ZooDB

Jonas Nick & Bogdan Vancea

May 29, 2014

Outline

- 1 Introduction
- 2 The new Index Implementation
- 3 Benchmarks

ZooDB

- an open source object database written in Java
- JDO standard compliant
- 4 times faster than competitor db4o
- zoodb.org

Database Index

Key-Value datastructure that allows for ordered iteration.

Database Index

Key-Value datastructure that allows for ordered iteration.

Example:

```
ZooJdoHelper.createIndex(pm, Person.class, "name",  
false);
```

Database Index

Key-Value datastructure that allows for ordered iteration.

Example:

```
ZooJdoHelper.createIndex(pm, Person.class, "name",  
false);
```

Attribute
Value \rightarrow Object-ID

Database Index

Key-Value datastructure that allows for ordered iteration.

Example:

```
ZooJdoHelper.createIndex(pm, Person.class, "name",  
false);
```

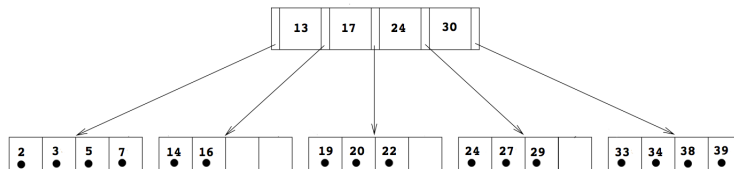
Attribute
Value \rightarrow Object-ID

OID
Object-ID
Diskpos

\rightarrow

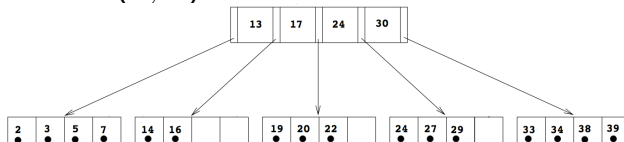
FSM
Page-ID \rightarrow TxID

B+ Tree

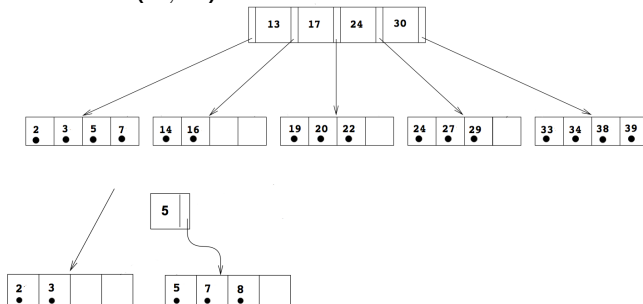


- node fills one disk page
- inner node contains keys and children pointer, leaves contain keys and values
- key unique vs. key-value unique

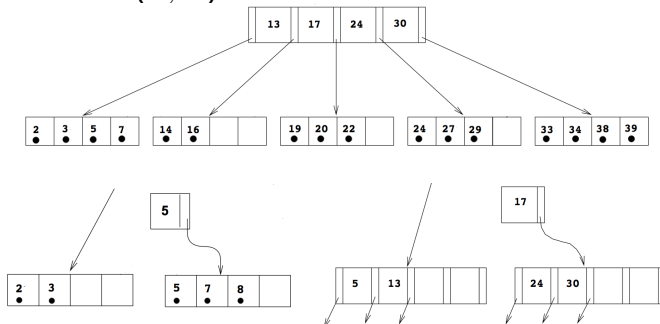
Example: insert (8, v)



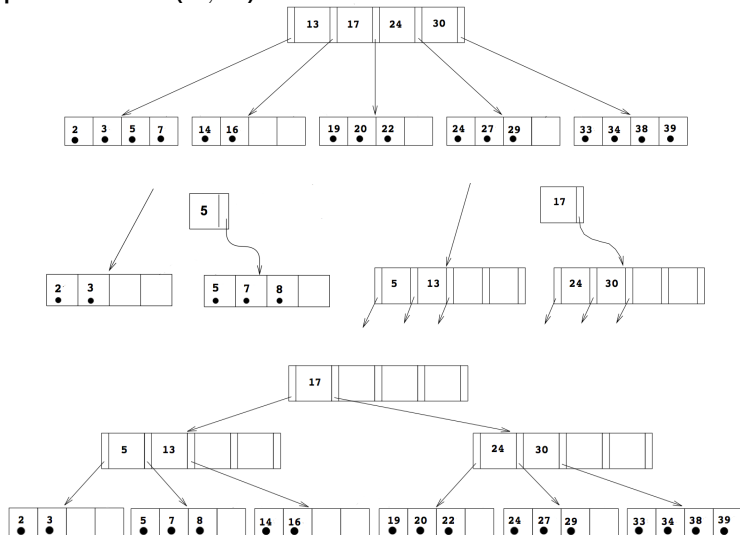
Example: insert (8, v)



Example: insert (8, v)



Example: insert (8, v)



3908 loc + 2665 loc of test

Goals

- fast B+ tree index
- buffer manager to allow caching
- prefix sharing

Challenges - General

- edge cases
- runtime dominated by disk access
 - change nodes infrequently
 - fewer nodes is better
- only parent to child pointer
- determine best when its time to split/redistribute
- key unique vs. key-value unique
- buffer manager lookup takes time
- prefix-sharing encoding/decoding takes time
- prefix-sharing rebalancing takes time
- general optimizations
 - avoid polymorphism
 - bit-level operations

Prefix Sharing

Exploit common prefix of keys in nodes

00010000

00010100

00010110

Prefix Sharing

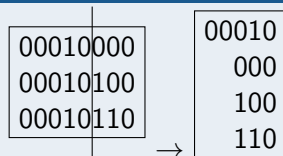
Exploit common prefix of keys in nodes

00010000
00010100
00010110

→

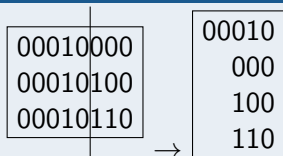
Prefix Sharing

Exploit common prefix of keys in nodes



Prefix Sharing

Exploit common prefix of keys in nodes

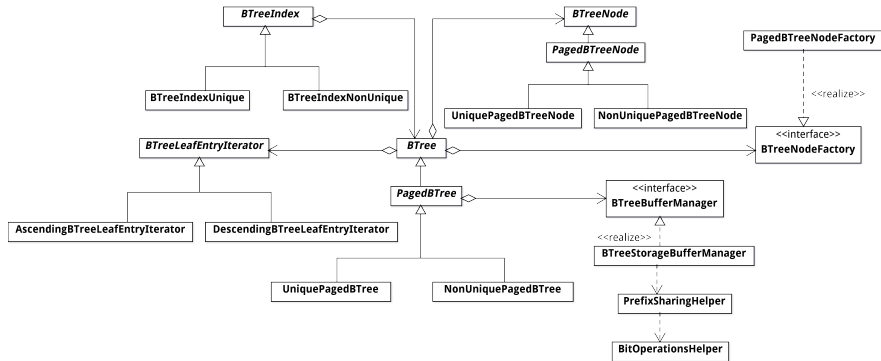


- variable number of key-value entries per page
- prefix determines
 - if 2 nodes can be merged without overflow
 - the number of entries that can be redistributed from one node to the other

Our B+ Tree

- 64 bit keys and 64 bit values
- key unique and key non-unique variants
- keys encoded based on common bit prefix

Class Diagram



Operations

- Search - Similar to normal B+ Tree
- Insert overflow
 - attempt to redistribute values to left sibling before creating a new node
- Delete underflow
 - check if possible to merge with left or right neighbour
 - check if possible to split current node between left and right
 - redistribute from left or right

Fine grained

- insert, remove, write
- duration, number of nodes
- prefix-sharing vs. no prefix-sharing

Whole system

- test harness
- partial PolePosition benchmark
- StackOverflow

Summary

- ...

Outlook

- ...