Indexes: Structure, Splits and Free Space Management Internals

Christian Antognini

@ChrisAntognini antognini.ch/blog

BASEL ▪ BERN ▪ BRUGG ▪ DÜSSELDORF ▪ FRANKFURT A.M. ▪ FREIBURG I.BR. ▪ GENEVA
HAMBURG ▪ COPENHAGEN ▪ LAUSANNE ▪ MUNICH ▪ STUTTGART ▪ VIENNA ▪ ZURICH

@ChrisAntognini

Senior principal consultant, trainer and partner at Trivadis

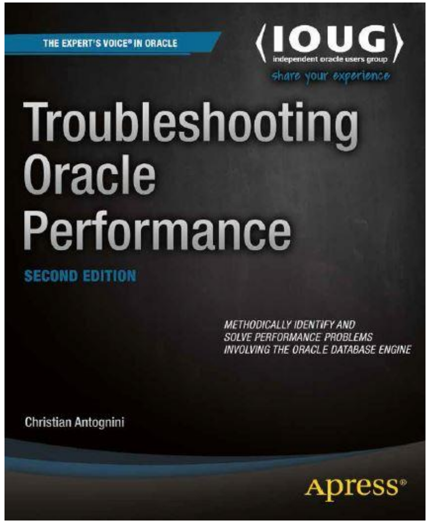
- christian.antognini@trivadis.com
- http://antognini.ch

Focus: get the most out of Oracle Database

- Logical and physical database design
- Query optimizer
- Application performance management

Author of Troubleshooting Oracle Performance (Apress, 2008/14)

OakTable Network, Oracle ACE Director


THE EXPERT'S VOICE® IN ORACLE

IOUG independent oracle users group share your experience

Troubleshooting Oracle Performance

SECOND EDITION

METHODICALLY IDENTIFY AND SOLVE PERFORMANCE PROBLEMS INVOLVING THE ORACLE DATABASE ENGINE

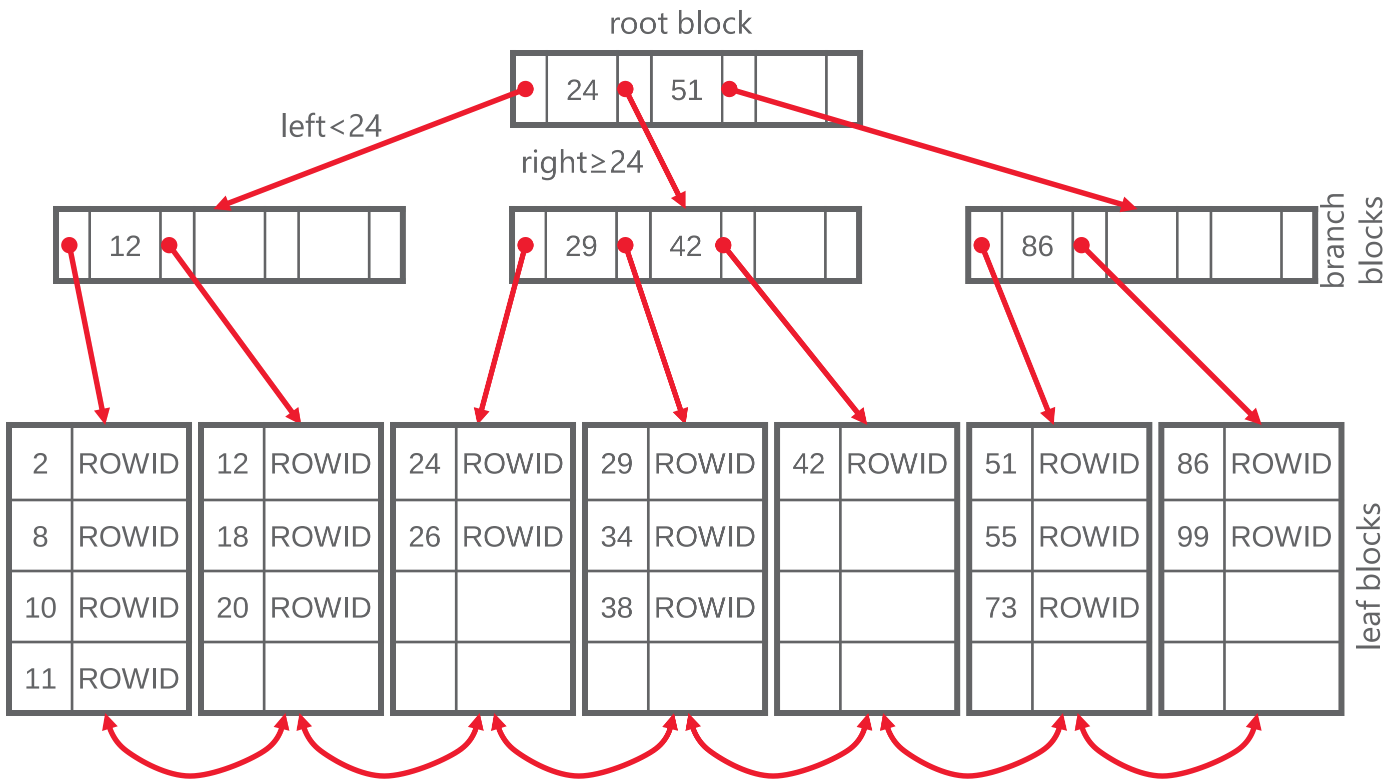
Christian Antognini

Apress®

Agenda

Concepts

Structure of a B-Tree



trivadis makes IT easier.

Nulls

NULL values are stored in the index only when at least one of the indexed columns is not NULL

- Exception: bitmap indexes always store indexed data

For unique indexes the uniqueness is guaranteed only if at least one of the indexed columns is not NULL

NULL values are stored after not NULL values in the index

Myths

B-tree indexes can become “unbalanced” over time and need to be rebuilt

Deleted space is “deadwood” and over time requires the index to be rebuilt

If an index reaches “x” number of levels, it becomes inefficient and requires the index to be rebuilt

If an index has a poor clustering factor, the index needs to be rebuilt

To improve performance, indexes need to be regularly rebuilt

Source: Oracle B-Tree Index Internals: Rebuilding The Truth, Richard Foote

Index Keys

Internal Key

Internally a unique key is needed to sort the indexed columns

For unique indexes the internal key is the same as the index key

For non-unique indexes the (extended or restricted) ROWID of the indexed row is added to the index key to make it unique

Therefore:

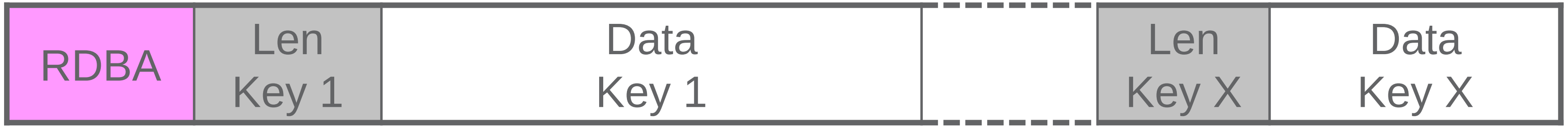
- The index entries are sorted in a consistent way
- Even for non-unique indexed the order of the index entries changes only when the indexed table is reorganized (i.e. not if the index is rebuilt!)

Branch Block Key

In branch blocks the internal key is truncated after the first byte that differs from the last key of the left leaf block

All keys are preceded by a length field

- 1 byte for data up to 127 bytes, otherwise 2 bytes
- Partially stored keys are terminated by 0xFE
- Nulls have a column length of 0xFF, trailing nulls <u>are</u> stored



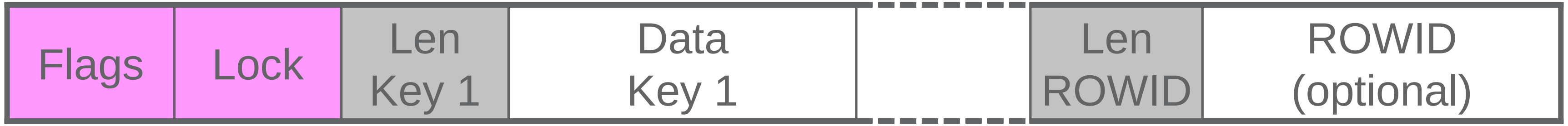
trivadis makes IT easier.

Leaf Block Key

All keys are preceded by a length field

- 1 byte for data up to 127 bytes, otherwise 2 bytes
- Nulls have a column length of 0xFF, trailing nulls <u>are</u> stored

For global indexes an extended ROWID is used, otherwise a restricted ROWID is enough.



trivadis makes IT easier.

Splits

Splits

50:50 split

- The keys are evenly distributed over two blocks
- Based on size, not number of keys

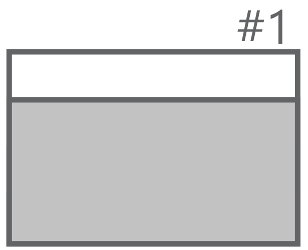
99:1 split (a.k.a. 90:10)

- The new key is the right-most of the index
- A new leaf block containing only the new key is added

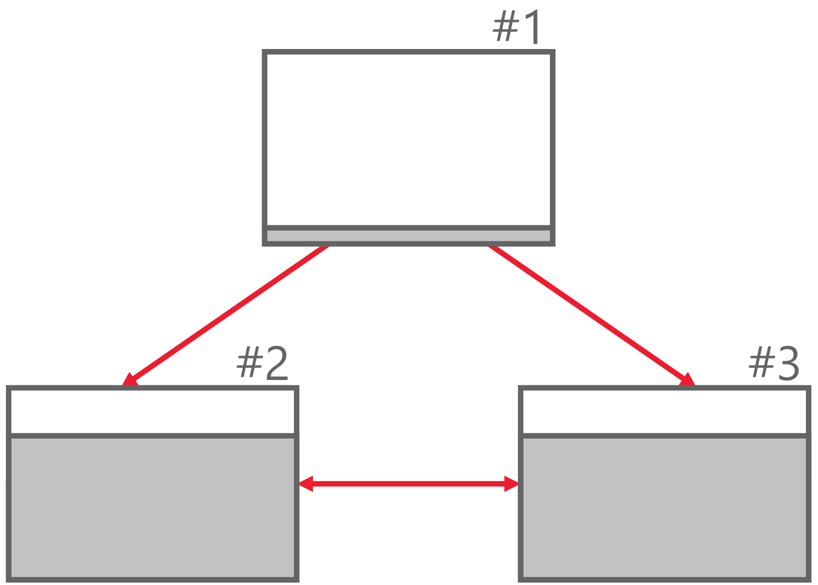
A change in the structure of an index (like a split) is not rollbacked if the transaction that caused it is rollbacked

trivadis makes IT easier.

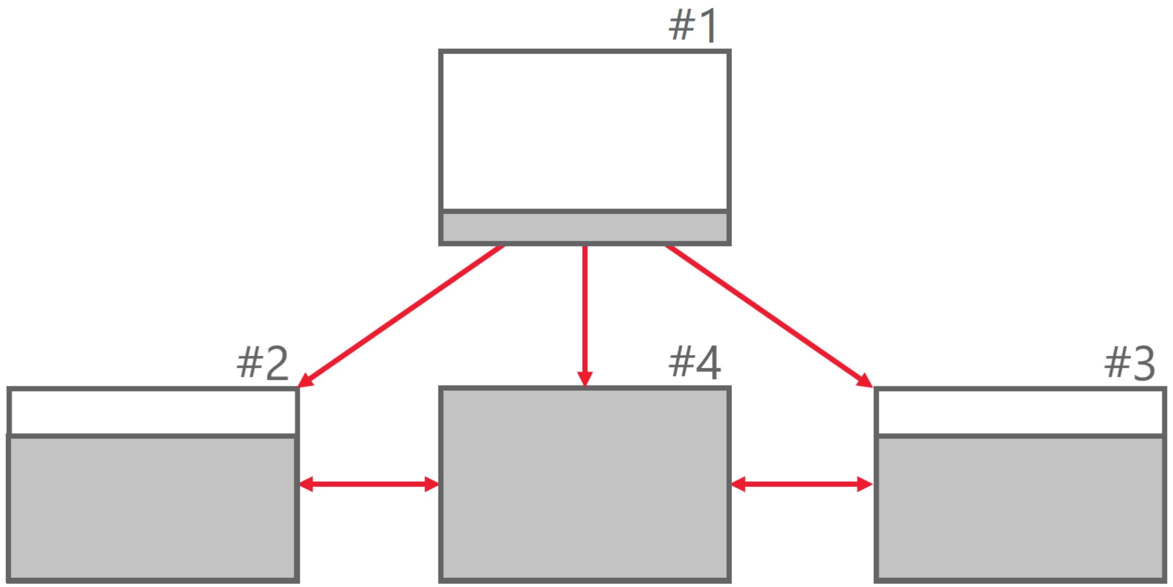
Splits



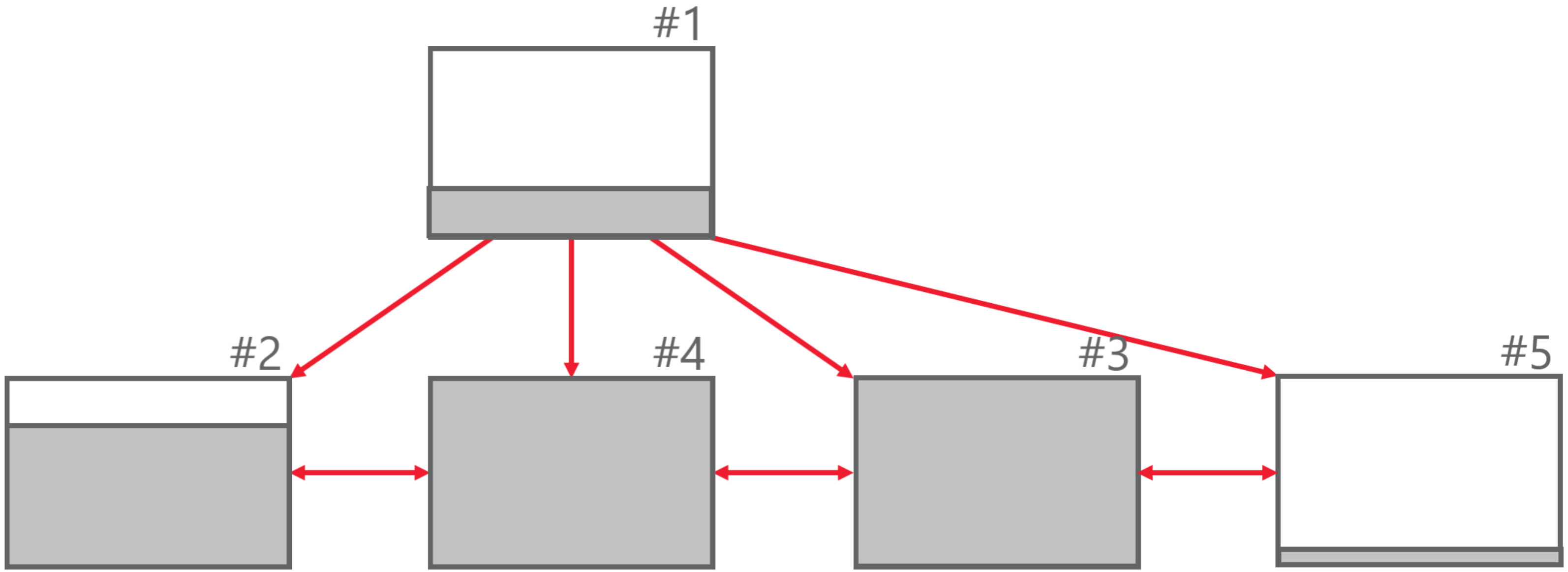
Splits



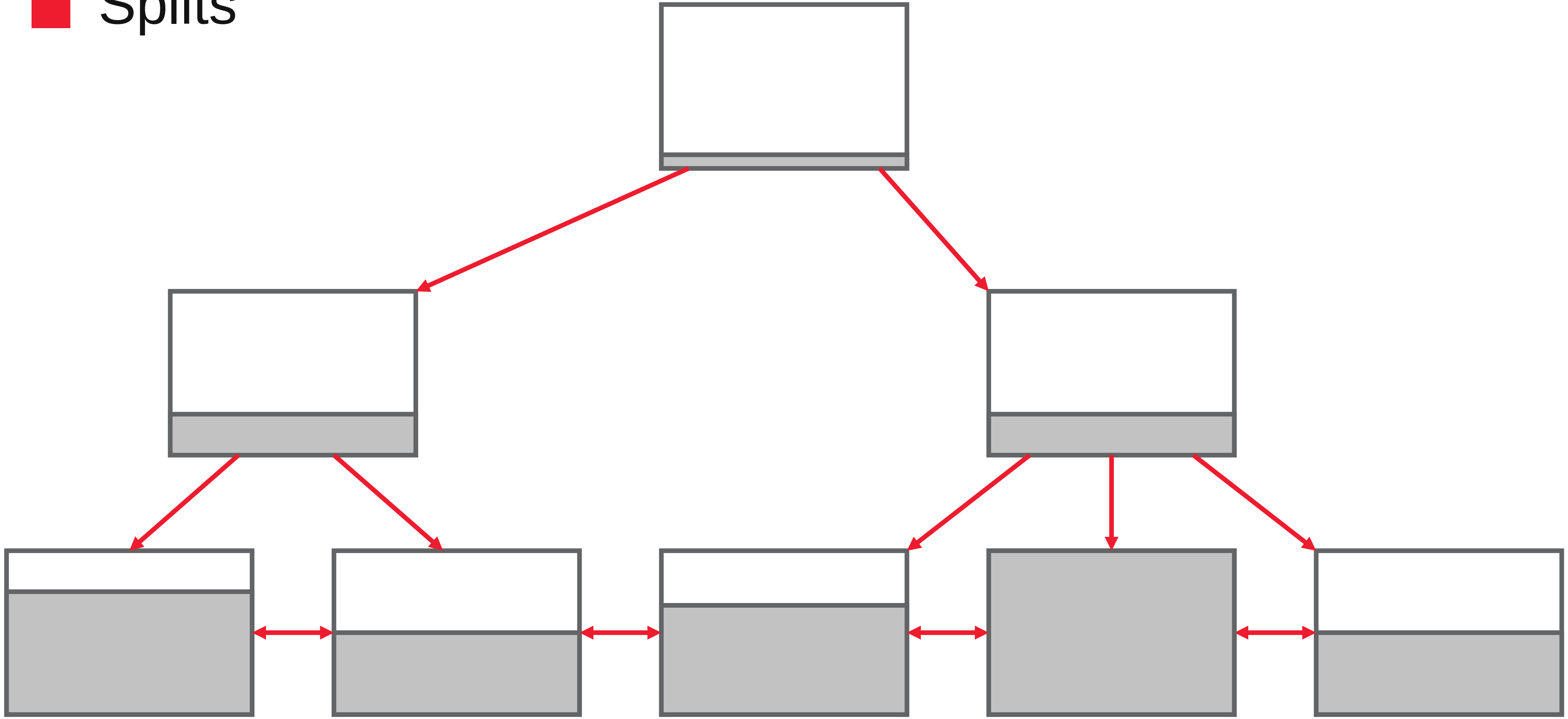
Splits



Splits



Splits



Free Space

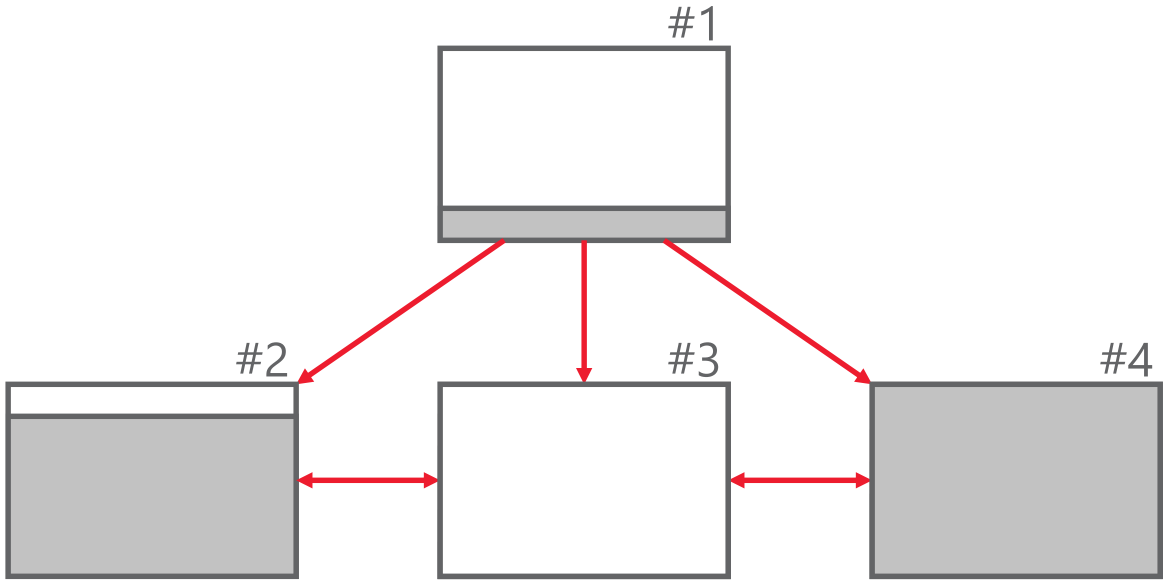
Reminder

PCTFREE is only used when the index is created (or rebuilt)

- An update is implemented as a delete followed by an insert
- The free space in leaf blocks is strongly dependent on the type of split (50:50 or 99:1)

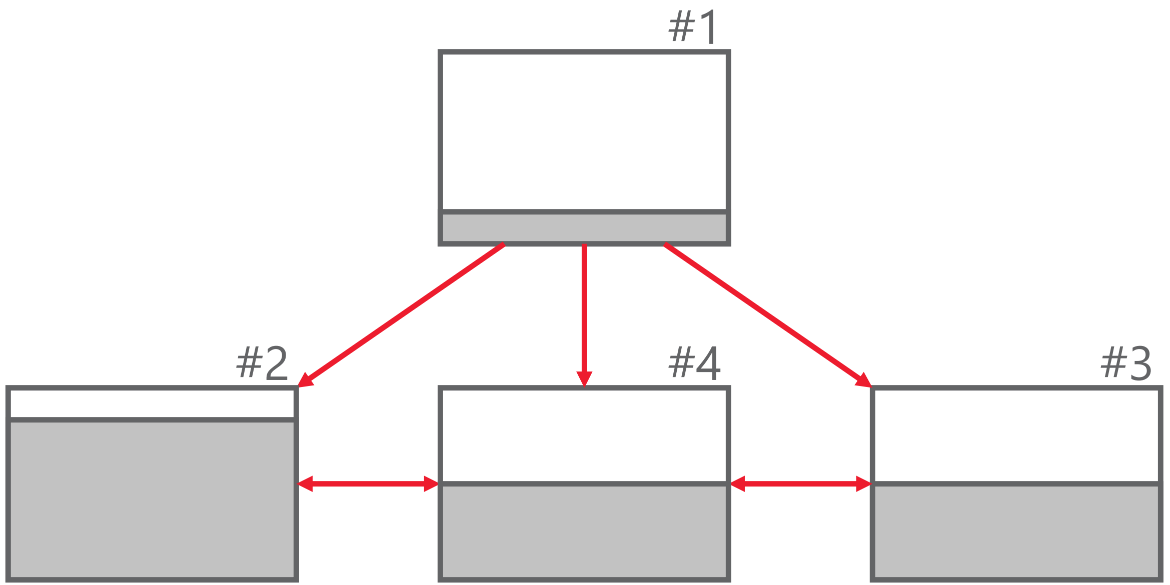
PCTUSED cannot be used for indexes

Free Space



trivadis makes IT easier.

Free Space



Reorganizations

Reorganizations

If the free space is not automatically reused, it should be manually reclaimed

- This is unusual, therefore such an operation should only be performed if it is really necessary

To manually reclaim free space two possibilities exists

- Rebuild: the index is recreated
- Coalesce: free space in contiguous blocks is merged and, therefore, some blocks are unchained

Rebuild vs. Coalesce

	Pro	Cons
Rebuild	▪ Doesn't lock table (online rebuilds only) ▪ Reclaim all the free space	▪ Locks table during rebuild (offline rebuilds only) ▪ Doubles space usage temporarily ▪ The index is completely recreated (to do so large sorts may be performed)
Coalesce	▪ Doesn't lock table ▪ Current index structure is reused	▪ Doesn't reclaim all the free space ▪ Can generate lot of redo

trivadis
makes IT easier.

Unnecessary Reorganizations

Unnecessary rebuild/coalesce may lead to sub-optimal data density in leaf blocks

- Too much free space, i.e. too low density
- Too many block splits, i.e. too high density

Bitmap

Bitmap

A bitmap index is a regular B-tree, only the internal key differs from the non-bitmap index!

The key is composed by:

- Indexed column(s)
- Start/End ROWID
- Bitmap representing the rows that have the key
  - – To save space it is “compressed”

Summary

- B-tree indexes cannot become "unbalanced"
- Deleted space in an index can be reused
- Indexes do not need to be regularly rebuilt

Questions and Answers

Christian Antognini
Senior Principal Consultant

christian.antognini@trivadis.com

@ChrisAntognini