



**UNIVERSITÄT  
DES  
SAARLANDES**

Faculty of Natural Sciences and Technology I  
Department of Computer Science

# Workload-based Data Partitioning for Index Construction

**Bachelor's Thesis**

written by

**Lukas Wilde**

**31st August 2022**

Supervisors

**Jens Dittrich**

Advisor

**First Advisor**

1st Reviewer

**Jens Dittrich**

2nd Reviewer

**Second Reviewer**



### **Eidesstattliche Erklärung**

Ich erkläre hiermit an Eides Statt, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

### **Statement in Lieu of an Oath**

I hereby confirm that I have written this thesis on my own and that I have not used any other media or materials than the ones referred to in this thesis.

### **Einverständniserklärung**

Ich bin damit einverstanden, dass meine (bestandene) Arbeit in beiden Versionen in die Bibliothek der Informatik aufgenommen und damit veröffentlicht wird.

### **Declaration of Consent**

I agree to make both versions of my thesis (with a passing grade) accessible to the public by having them added to the library of the Computer Science Department.

Saarbrücken,

\_\_\_\_\_  
Datum/Date

\_\_\_\_\_  
Unterschrift/Signature

# Acknowledgement

# Abstract



# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Related Work</b>	<b>2</b>
<b>3</b>	<b>Background</b>	<b>3</b>
3.1	Hybrid Index Structures . . . . .	3
3.2	Partition and Partitioning functions . . . . .	3
3.3	Numerical Differentiation . . . . .	3
<b>4</b>	<b>Approach and Algorithms</b>	<b>4</b>
4.1	Approach . . . . .	4
4.2	Partitioning by Frequency . . . . .	4
4.3	Partitioning by Purity . . . . .	4
<b>5</b>	<b>Datasets and Workloads</b>	<b>5</b>
5.1	Datasets . . . . .	5
5.1.1	Synthetic Datasets . . . . .	5
5.1.2	Real-world Datasets . . . . .	5
5.2	Workloads . . . . .	5
5.2.1	Synthetic Workloads . . . . .	5
5.2.2	Real-world Workloads . . . . .	5
<b>6</b>	<b>Evaluation</b>	<b>7</b>
6.1	Setup . . . . .	7
6.2	Lookup Performance . . . . .	7
6.2.1	Frequency Algorithm . . . . .	7
6.2.2	Purity Algorithm . . . . .	7
6.3	Role of Partitioning Parameters . . . . .	7
6.3.1	Frequency Algorithm . . . . .	7
6.3.2	Purity Algorithm . . . . .	7
<b>7</b>	<b>Conclusion and Future Work</b>	<b>9</b>
<b>A</b>	<b>Appendix</b>	<b>i</b>





# Chapter 1

## Introduction

Here is a citation [1].

- DBMS routinely use index structures for increased performance
- Index pre-configured or chosen by user
- Mostly no utilization of underlying data or workload distribution
- Except: learned indexes -> Related Work
- Motivation: different data structures for different query workloads (hash table?)
- For this, introduce concept of hybrid index structures
- Create partitions to create singular indexes and combine them
- Optimize partitions based on one/multiple metrics
- As motivation: GENE, starting point for generic search
- Introduce what is covered in what section of this thesis

## Chapter 2

# Related Work

In this chapter, I present work related to the topic of workload-based data partitioning. Additionally, we reference work that deals with data partitioning in a broader sense.

### 1. *Baseline indexes*

- B-tree [2]
- ART index [3]

### 2. *Hybrid/Adaptive Index Structures*

- Adaptive Hybrid Indexes [4] for context and compacting/decompressing criterias
- GENE [5] for the approach to look at indexes as logical components and combining them, generic search briefly to iterate over starting options and give our partitioning as possible better starting point.

### 3. *Learned Index Structures*

Emphasize how they use the underlying data distribution

- FITing-Tree [6] as introduction to PGM (brief)
- PGM-index [7] for optimal linear piecewise partitioning
- RMI/ALEX [8], [9] with their tree-like model structure, ALEX to improve upon caveats of RMI (updatability)

### 4. *Distributed (Database) Systems*

Context: data partitioning in the sense that different partitions can be stored on different nodes of the distributed system

- Schism [10] for their workload-centric approach to data partitioning

# Chapter 3

## Background

### 3.1 Hybrid Index Structures

- What are hybrid index structures?
- Advantages: optimize for subproblems, combine to one index
- challenges: correct combination of these structures (e.g. routing through data structure)

### 3.2 Partition and Partitioning functions

- Mathematical set theory definition of partition
- Adaption to key space/segments
- Partitioning functions for indexes
- Used in routing of through index

### 3.3 Numerical Differentiation

- Finite difference approximations
- Relation to true derivative ( $\lim_{h \rightarrow 0}$ )
- Consistency order of approximations
- Forward, Central, Backward finite difference approximations

## Chapter 4

# Approach and Algorithms

### 4.1 Approach

- Data generation
- Workload generation + parameters
- Partitioning (more details in section 4.2 and 4.3)
- Interface between Partitioning and Bulkloading
- (Informed) Bulkloading
- Benchmarking

### 4.2 Partitioning by Frequency

- Motivation: caching
- Idea from numerical approximations
- Algorithm

### 4.3 Partitioning by Purity

- Motivation: optimize index for different query types
- Algorithm

# Chapter 5

## Datasets and Workloads

This chapter deals with the used datasets and workloads

### 5.1 Datasets

- Generation procedure
- Used parameters for parameterized distributions

#### 5.1.1 Synthetic Datasets

- uniform dense

#### 5.1.2 Real-world Datasets

- SOSD datasets (osm, books, fb)

### 5.2 Workloads

#### 5.2.1 Synthetic Workloads

- uniform sampling
- lognormal (because used in hybrid adaptive indexing paper)
- step workloads
- Proof of concept workload

#### 5.2.2 Real-world Workloads

- Self-generated
- Are they representative (look into dbbench/YSCB)

- workloads especially OLTP often skewed (Identifying Hot and Cold Data in Main-Memory Databases, <https://www.microsoft.com/en-us/research/wp-content/uploads/2013/04/ColdDataClassification-icde2013-cr.pdf>)

# Chapter 6

## Evaluation

This chapter will deal with the evaluation of the experiments

### 6.1 Setup

- hardware
- index parameters like slot size, PGM epsilon etc.

### 6.2 Lookup Performance

#### 6.2.1 Frequency Algorithm

#### 6.2.2 Purity Algorithm

### 6.3 Role of Partitioning Parameters

- $window\_size$   
, delta for frequency
- $window\_size$   
for purity (as of yet)

#### 6.3.1 Frequency Algorithm

#### 6.3.2 Purity Algorithm





## Chapter 7

# Conclusion and Future Work

- Previous results reproducible?
- What have we found?
- Does partitioning yield better lookup times?
- Is it beneficial to move leaves higher up in tree?
- Is it beneficial to use hybrid index structures (i.e. change layout/data structure in nodes)
- Best case/worst case considerations?
- Future Work: Combination of metrics
- Future Work: Look at more data structures other than BinarySearch-Leaves and Hashtables
- Future Work: What other workload metrics can be used for partitioning?



# Bibliography

- [1] Douglas Comer. ‘Ubiquitous B-Tree’. In: *ACM Comput. Surv.* 11.2 (June 1979), pp. 121–137. ISSN: 0360-0300. DOI: 10.1145/356770.356776. URL: <https://doi.org/10.1145/356770.356776>.
- [2] R Bayer and E McCreight. ‘Organization and maintenance of large ordered indices’. In: *Proceedings of the 1970 ACM SIGFIDET (now SIGMOD) Workshop on Data Description, Access and Control - SIGFIDET '70*. Houston, Texas: ACM Press, 1970.
- [3] Viktor Leis, Alfons Kemper and Thomas Neumann. ‘The adaptive radix tree: ARTful indexing for main-memory databases’. In: *2013 IEEE 29th International Conference on Data Engineering (ICDE)*. 2013, pp. 38–49. DOI: 10.1109/ICDE.2013.6544812.
- [4] Christoph Anneser et al. ‘Adaptive Hybrid Indexes’. In: *Proceedings of the 2022 International Conference on Management of Data*. ACM, June 2022. DOI: 10.1145/3514221.3526121. URL: <https://doi.org/10.1145/3514221.3526121>.
- [5] Jens Dittrich, Joris Nix and Christian Schön. ‘The next 50 years in database indexing or’. In: *Proceedings of the VLDB Endowment* 15.3 (Nov. 2021), pp. 527–540. DOI: 10.14778/3494124.3494136. URL: <https://doi.org/10.14778/3494124.3494136>.
- [6] Alex Galakatos et al. ‘FITing-Tree’. In: *Proceedings of the 2019 International Conference on Management of Data*. ACM, June 2019. DOI: 10.1145/3299869.3319860. URL: <https://doi.org/10.1145/3299869.3319860>.
- [7] Paolo Ferragina and Giorgio Vinciguerra. ‘The PGM-index: a fully-dynamic compressed learned index with provable worst-case bounds’. In: *PVLDB* 13.8 (2020), pp. 1162–1175. ISSN: 2150-8097. DOI: 10.14778/3389133.3389135. URL: <https://pgm.di.unipi.it>.
- [8] Tim Kraska et al. *The Case for Learned Index Structures*. 2017. DOI: 10.48550/ARXIV.1712.01208. URL: <https://arxiv.org/abs/1712.01208>.

- [9] Jialin Ding et al. ‘ALEX: An Updatable Adaptive Learned Index’. In: *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*. ACM, June 2020. DOI: 10.1145/3318464.3389711. URL: <https://doi.org/10.1145/3318464.3389711>.
- [10] Carlo Curino et al. ‘Schism’. In: *Proceedings of the VLDB Endowment* 3.1-2 (Sept. 2010), pp. 48–57. DOI: 10.14778/1920841.1920853. URL: <https://doi.org/10.14778/1920841.1920853>.

Appendix A

Appendix