Database Systems and Information Management Group
Fak. IV Electrical Engineering and Computer Science
Technische Universität Berlin

# Thesis Proposal

**Instructions to Students**

(a) Complete the proposal below and ask your primary advisor to review it for completeness.
(b) Once it has been accepted, your primary advisor should forward it to Juan Soto for final approval.
(c) The proposal length **must not** exceed *five* pages, including the cover and bibliography pages.

| | | | |
|---|---|---|---|
| **Thesis Type** | Master's | **ECTS** | 30 |
| **Student** | Riccardo Marin | **Mat. Number** | 476629 |
| **Email** | riccardo.marin@campus.tu-berlin.de | **Primary Advisor** | Varun Pandey |
| **1st Examiner** | Prof. Dr. Volker Markl | **2nd Examiner** | - |
| **Academic Program** | Computer Science | | |
| **Thesis Title** | *Towards an Optimal Physical Layout for Efficient Query Processing in the Cloud* | | |
| **Duration Period** | From 09/2023 to 02/2024 | | |

## 1. Introduction / Scientific Background / Related Work

### *1.1* Specify the *obstacle* to be overcome

Multi-dimensional data partitioning is mainly used in analytical data processing, where users perform range queries over several dimensions. However, each multi-dimensional structure has different properties and trade-offs. The main challenge is finding the optimal data structure for query processing, depending on the workload.

### *1.2* Specify the *motivation* (Why is this problem interesting?)

Major cloud-native warehouses follow a shared-nothing architecture effectively separating compute from storage. To improve data access, and consequently query performance, these data warehouses maintain coarse-grained column statistics based indexes (such as min-max per column) to fetch partitions relevant to the query [9]. These indexes, also known as Zone Maps [2], perform inadequately on unordered data. Gradually, many data warehouses have added support for statistics computation and data clustering to facilitate efficient data access [3].
The combination of several factors – such as the query selectivity, the presence of dimension ordering, the data distribution – determines which is the most performant physical layout for the query. For example, ordering on one dimension does not benefit queries that select other unordered columns. On the other hand, techniques to index or partition data along several dimensions suffer from the curse of dimensionality. There are also a number of well-known multiple-dimensional indexes, resulting in a greater number of data partitioning and ordering options. However, it is unclear which of these partitioning algorithms operates best under which workloads.

### 1.3 Specify the *novelty* (Was this problem already solved?)

A popular approach is creating a suitable partitioning for a given query workload [13]. These partitions may also change and adapt as the workload varies. Most of these techniques do not scale to higher dimensions [12]. In fact, several partitioning techniques for multidimensional data, e.g., k-d trees, R-trees, and quadtrees are typically used for spatial data with two dimensions. Nevertheless, a comprehensive analysis for partitioning on multiple dimensions is missing. In particular, we want to shed light on the benefits and limitations of several multidimensional partitioning approaches across different datasets and query distribution. The thesis aims at closing this gap.

### 1.4 Specify the *anticipated impact* (how does solving this problem impact our world?)

The resulting benchmarks will provide a detailed understanding of the behavior of several multidimensional partitioning techniques with a variety of workloads and settings. This information can contribute to the choice of the optimal physical layout, by directly applying the obtained knowledge or used as a building block for a learned data layout (e.g. in the training phase).

### 1.5 Specify the *anticipated contributions* (*journal publication*? *open-source software*?)

Implementation in C++ of 7 multi-dimensional structures – Fixed-Grid, Grid-file, kd-tree, Quadtree, STRTree, Z-ordering, Hilbert curve and read/write/partitioning of data into Parquet files based on such structures. Design of a set of queries to use in the measurements. Benchmarks computing the numbers of partitions retrieved by the queries across different structures, queries and partition size. The source code and the results will be made available on a public GitHub repository.

### 1.6 Specify the *scientific background* (or foundational work) that your solution will build upon

Effective data partitioning leads to improved performance for modern data warehouses.
The proposed solution will utilize several well-known multidimensional partitioning techniques to partition the data. The partitions will be stored in a widely-used file format from the open-source community, Parquet [19]. This file format has the advantage of maintaining a set of statistics in the file metadata (max, min, avg) for each dimension. The metadata is used to prune the files to fetch and therefore obtain only the partitions containing the queried data. In order to measure the end-to-end performance, we will use DuckDB as the base database system, which has support to push down predicates to Parquet.

### 1.7 Specify the *related work* (e.g., competing solutions corresponding to the research challenge)

An extensive survey of multidimensional access methods [1] for spatial datasets [5] will be considered as a starting point for the research. The foundational techniques used on partitioned data, including data clustering and zone maps are covered by several publications [2] [3]. Since the workload and data distribution determine how efficient is the retrieval of the matching tuples, adaptive approaches have been proposed. One of the first non-static approaches was database cracking [8]: an index is built as a side product of query processing. More recently, workload-based adaptive systems dynamically change the partitions without needing an upfront workload [10, 12]. Amoeba [10] builds an upfront partitioning tree based on all attributes and the adaptive query executor updates the tree according to the incoming queries. It also handles possible imbalances in the tree generated by correlated or skewed data. Yang et al propose the Qd-Tree, a data structure similar to a kd-tree, where each node contains a predicate expression that splits the physical data space, in order to achieve the maximum block skipping [17]. Pando [14] extends this idea by looking for correlations between the results of the different logical predicates in the queries. In Schism [18] the clustering of data is done by observing the correlation between the tuples within the transactions. Each tuple is mapped to a node of a graph and the graph is divided into partitions. There is also a line of work that takes into account both data distribution and workload [15, 16]. A

similar approach is applied in adaptive indexes, which employ machine learning techniques to determine the optimal index [4], such as in Flood [6] and Tsunami [7] projects.

## 2. Goal of the Thesis / Statement of the Research Problem

**2.1 State the overarching goal that you aim to achieve in your thesis.**
The goal is to quantify the performance of several multidimensional partitioning techniques and evaluate how the conditions (such as data distribution, query workload, partition size) affect the result.

**2.2 Provide a succinct, precise, and unambiguous *statement of the research problem(s)* or *question* to be solved, in order to attain the goal.**
When dealing with range queries in cloud-based data warehouses, how do the different partitioning techniques compare? Under what conditions does a partitioning based on a one-dimensional structure outperform a multi-dimensional one?

**2.3 Specification of the *scope of the thesis*.**

**2.3.1 State what is *in scope*, particularly, the *subproblems that will be explored.***
The scope includes data partitioning techniques in data warehousing, with a focus on the different multi-dimensional structures.

**2.3.2 State what is *out of scope*.**
Data streams, graph databases, transaction management, modern hardware, query compilation.

## 3. Thesis Approach

**3.1 Describe the solution approach (e.g., algorithms, data, evaluation metrics, software, systems).**

| Student_id | Age |
|---|---|
| 16 | 30 |
| 45 | 21 |
| 21 | 18 |
| 7 | 27 |
| 74 | 41 |
| 34 | 37 |
| 111 | 23 |
| 91 | 22 |

Figure 1

A motivating example is the following. We assume a context with a dataset consisting of two columns, *student_id* and *age* (Figure 1). We consider three different approaches for the partitioning strategy: (a) clustering on one dimension, (b) kd-tree and (c) QuadTree (Figure 2). They respectively induce 4, 4 and 5 partitions, with the data distributed in different ways (Figure 3).

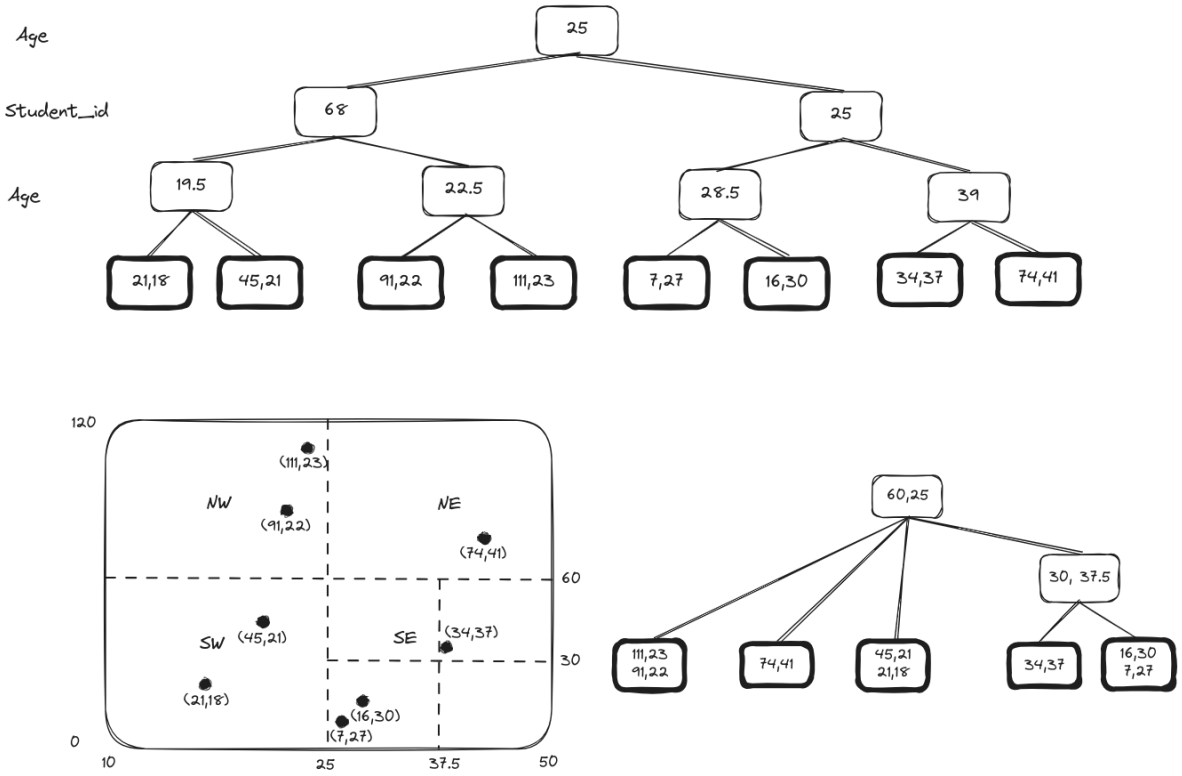


Figure 2

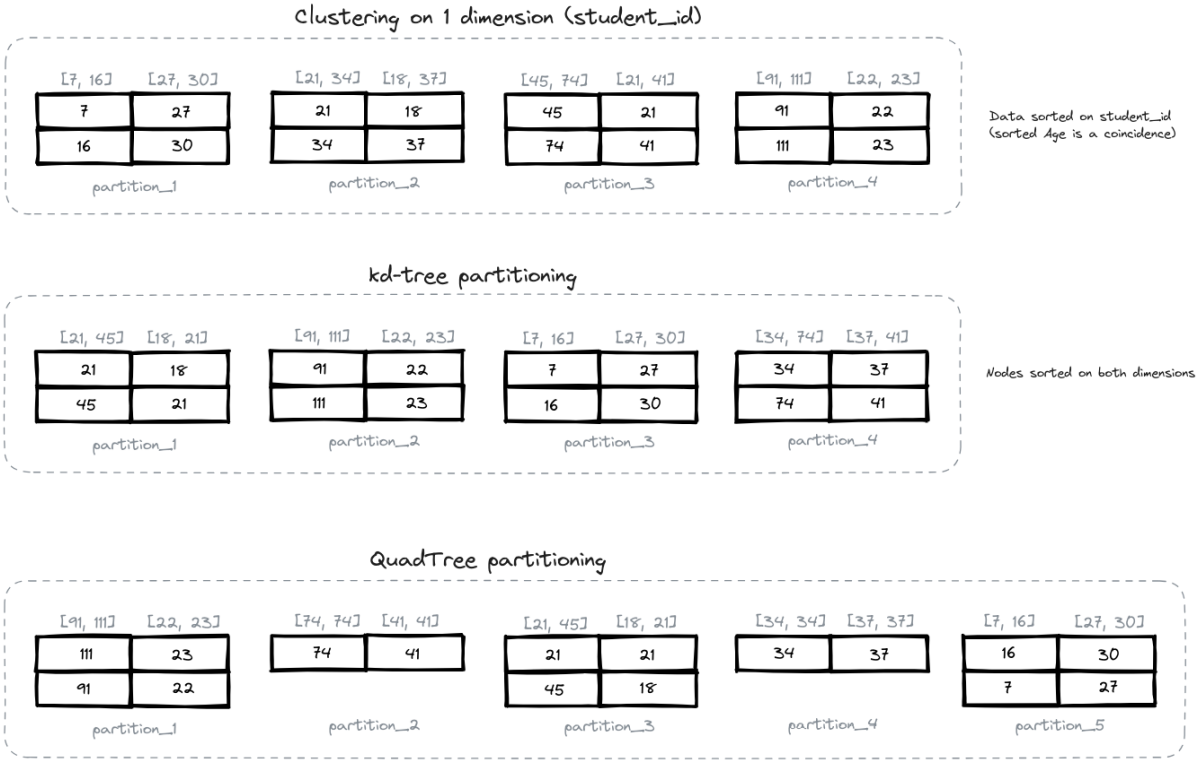


Figure 3

We use two example queries to compare them: Q1: selects all the columns where *student_id* falls in range 40-70 and age is between 25 and 35; Q2: similar to Q1 but filters only one attribute, *student_id* between 10 and 40. The approach (a) fetches 3 partitions for Q1 and 2 for Q2, showing better performance in the latter. Conversely, (b) and (c) retrieve 1 partition for Q1 - demonstrating an high pruning power for queries on multiple dimensions - and 3 for Q2 (Figure 4, 5).

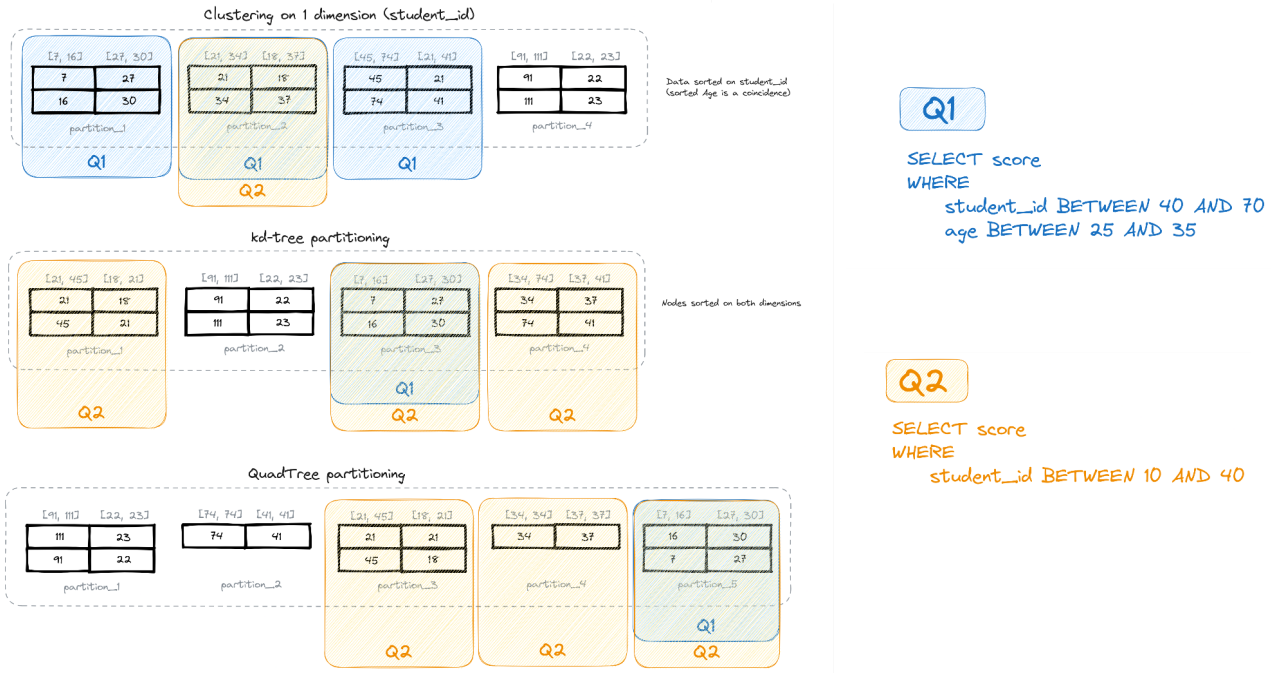


Figure 4

| Approach | Q1 | Q2 |
|---|---|---|
| Clustering s_id | 3 | 2 |
| kd-tree | 1 | 3 |
| QuadTree | 1 | 3 |

Figure 5

The artifacts will be implemented in C++. The multidimensional structures will include: Fixed-Grid, Grid-file, kd-tree, Quadtree, STRTree, Z-curve, Hilbert curve. A module for reading and writing Parquet partitions based on these structures will be included as well. Several datasets of relevant size with a high number of dimensions can be used in the benchmarks. Those open source and used in Flood [6] and Tsunami [7] will be considered: OpenStreetMap [20], Taxi [21], TPC-H [22] and Stocks [23]. A set of relevant queries will be created. DuckDB will be used to evaluate the performance of the different partitioning techniques. The benchmarks will measure the number of partitions used in the queries across different selectivity, number of columns and partition size.

### 3.2 How does the proposed solution differ from the state-of-the-art?

Multiple studies address the problem of finding the optimal data layout [10, 12, 14, 15, 17, 18], but they lack a focus on multidimensional workloads. Several recent papers involve the use of machine learning techniques to find the optimal structure [11], but they also do not include an in-depth performance analysis of the possible multi-dimensional techniques. This solution can both better

assess the behavior of the partitioning strategies across workloads and be used as a source of labeled data for machine learning training.

**3.3 How will you know if you have *succeeded* in attaining your goal, i.e., solved the research problem(s)?**

**3.3.1 How will you measure the *effectiveness* of your solution?**

There will be a set of reproducible benchmarks running on DuckDB, measuring the number of partitions used to answer the queries. The results clearly highlight how in different settings the multi-dimensional approach performance differs from the one-dimensional techniques.

**3.3.2 How and against which baseline(s) will you measure the *efficiency* of your solution?**

The main metric will be the number of partitions fetched for answering a single query. The measurements will be conducted across a set of queries, varying selectivity and filtered number of columns. 7 approaches will be benchmarked and compared.

## 4. Implementation Plan and Timeframe

| Dates | Milestone | Tasks | Deliverable(s) |
|---|---|---|---|
| 01.09 – 15.09 | MS1 | 1. conduct literature review | list of relevant papers |
| | | 2. read through the identified papers | reading notes |
| 16.09 – 31.09 | MS2 | 3. gain familiarity with multi-dimensional and one-dimensional structures | reading notes |
| | | 4. design the system architecture | architectural scheme |
| 01.10 – 31.11 | MS3 | 5. implement partitioning in Parquet | C++ implementation |
| | | 6. implement FixedGrid | C++ implementation |
| | | 7. implement Grid-File | C++ implementation |
| | | 8. implement kd-Tree | C++ implementation |
| | | 9. implement QuadTree | C++ implementation |
| | | 10. implement STRTree | C++ implementation |
| | | 11. implement Z-curve | C++ implementation |
| | | 12. implement Hilbert curve | C++ implementation |
| | | 13. validate prototype/system effectiveness | unit tests |
| 01.12 – 14.01 | MS4 | 14. define data, workload(s), evaluation metrics | benchmark settings: data, queries and metrics |
| | | 15. design experiments to be conducted | benchmark protocol |
| | | 16. conduct experiments | benchmark data |
| 15.01 – 31.01 | MS5 | 17. analyze/interpret results (explain anomalies) | benchmark plots and evaluation |
| 01.02 – 28.02 | TC[1] | 18. complete thesis writeup | submit thesis by the 28.02[2] |

[1] thesis complete

[2] add an approximate date, the precise due date will be provided by the Prüfungsamt in writing at a later date.

| | | | |
|---|---|---|---|
| 15.02 – 28.02 | **SC[3]** | 19. finalize prototype/system | publish code in an open repository |

**Preparing for the Thesis Defense**

*Students should sign and date that they have read the instructions below.*

| Date | Tasks | General Instructions |
|---|---|---|
| ***post thesis submission*** | 20. prepare presentation slides | **send slides (as a .pdf file) to advisor(s) and Juan**, no later than 8:00 a.m. (GMT+1), the day of the scheduled defense |
| ***defense date*** | 21. deliver the presentation | Time limited: 15' for B.Sc. and 20' for M.Sc. |
| | 22. answer raised questions | ca. 20' for Q&A |

**Signature:** Riccardo Marin **Date: 20.09.2023**

## 5. Bibliography

***Instructions.*** Use the *ACM Bibliography Style* (https://www.bibtex.com/s/bibliography-style-base-acm/) to list the set of reference sources you drew on to prepare your thesis proposal.


[1] Hanan Samet. 2005. *Foundations of Multidimensional and Metric Data Structures* (The Morgan Kaufmann Series in Computer Graphics and Geometric Modeling). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[2] Guido Moerkotte. 1998. *Small Materialized Aggregates: A Light Weight Index Structure for Data Warehousing*. In Proceedings of the 24rd International Conference on Very Large Data Bases (VLDB '98). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 476–487.

[3] Mohamed Ziauddin, Andrew Witkowski, You Jung Kim, Dmitry Potapov, Janaki Lahorani, and Murali Krishna. 2017. *Dimensions based data clustering and zone maps*. Proc. VLDB Endow. 10, 12 (August 2017), 1622–1633. https://doi.org/10.14778/3137765.3137769

[4] Pandey, Varun, Alexander van Renen, Andreas Kipf, Ibrahim Sabek, Jialin Ding and Alfons Kemper. 2020. *The Case for Learned Spatial Indexes*. ArXiv abs/2008.10349.

[5] Volker Gaede and Oliver Günther. 1998. *Multidimensional access methods.* ACM Comput. Surv. 30, 2 (June 1998), 170–231. https://doi.org/10.1145/280277.280279

[6] Vikram Nathan, Jialin Ding, Mohammad Alizadeh, and Tim Kraska. 2020. *Learning Multi-Dimensional Indexes.* In Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data (SIGMOD '20). Association for Computing Machinery, New York, NY, USA, 985–1000. https://doi.org/10.1145/3318464.3380579

[7] Jialin Ding, Vikram Nathan, Mohammad Alizadeh, and Tim Kraska. 2020. *Tsunami: a learned multi-dimensional index for correlated data and skewed workloads*. Proc. VLDB Endow. 14, 2 (October 2020), 74–86. https://doi.org/10.14778/3425879.3425880

[8] Idreos, Stratos & Kersten, Martin & Manegold, Stefan. (2007). *Database Cracking.* Journal of The American Society for Mass Spectrometry - J AMER SOC MASS SPECTROM. 68-78.

[9] Oshrit Feder, Guy Khazma, Gal Lushi, Yosef Moatti, and Paula Ta-Shma. 2019. *Big data skipping in the cloud.* In Proceedings of the 12th ACM International Conference on Systems and Storage (SYSTOR


[3] system complete

'19). Association for Computing Machinery, New York, NY, USA, 193. https://doi.org/10.1145/3319647.3325854
[10] Anil Shanbhag, Alekh Jindal, Samuel Madden, Jorge Quiane, and Aaron J. Elmore. 2017. *A robust partitioning scheme for ad-hoc query workloads.* In Proceedings of the 2017 Symposium on Cloud Computing (SoCC '17). Association for Computing Machinery, New York, NY, USA, 229–241. https://doi.org/10.1145/3127479.3131613
[11] Jun Rao, Chun Zhang, Nimrod Megiddo, and Guy Lohman. 2002. *Automating physical database design in a parallel database.* In Proceedings of the 2002 ACM SIGMOD international conference on Management of data (SIGMOD '02). Association for Computing Machinery, New York, NY, USA, 558–569. https://doi.org/10.1145/564691.564757
[12] Ahmed M. Aly, Ahmed R. Mahmood, Mohamed S. Hassan, Walid G. Aref, Mourad Ouzzani, Hazem Elmeleegy, and Thamir Qadah. 2015. *AQWA: adaptive query workload aware partitioning of big spatial data.* Proc. VLDB Endow. 8, 13 (September 2015), 2062–2073. https://doi.org/10.14778/2831360.2831361
[13] Jindal, A., Shanbhag, A., Lu, Y. (2019). *Robust Data Partitioning.* In: Sakr, S., Zomaya, A.Y. (eds) Encyclopedia of Big Data Technologies. Springer, Cham. https://doi.org/10.1007/978-3-319-77525-8_149
[14] Sivaprasad Sudhir, Wenbo Tao, Nikolay Laptev, Cyrille Habis, Michael Cafarella, and Samuel Madden. 2023. *Pando: Enhanced Data Skipping with Logical Data Partitioning*. Proc. VLDB Endow. 16, 9 (May 2023), 2316–2329. https://doi.org/10.14778/3598581.3598601
[15] Jialin Ding, Umar Farooq Minhas, Badrish Chandramouli, Chi Wang, Yinan Li, Ying Li, Donald Kossmann, Johannes Gehrke, and Tim Kraska. 2021. *Instance-Optimized Data Layouts for Cloud Analytics Workloads.* In Proceedings of the 2021 International Conference on Management of Data (SIGMOD '21). Association for Computing Machinery, New York, NY, USA, 418–431. https://doi.org/10.1145/3448016.3457270
[16] Tim Kraska. 2021. *Towards instance-optimized data systems*. Proc. VLDB Endow. 14, 12 (July 2021), 3222–3232. https://doi.org/10.14778/3476311.3476392
[17] Zongheng Yang, Badrish Chandramouli, Chi Wang, Johannes Gehrke, Yinan Li, Umar Farooq Minhas, Per-Åke Larson, Donald Kossmann, and Rajeev Acharya. 2020. *Qd-tree: Learning Data Layouts for Big Data Analytics.* In Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data (SIGMOD '20). Association for Computing Machinery, New York, NY, USA, 193–208. https://doi.org/10.1145/3318464.3389770
[18] Carlo Curino, Evan Jones, Yang Zhang, and Sam Madden. 2010. *Schism: a workload-driven approach to database replication and partitioning.* Proc. VLDB Endow. 3, 1–2 (September 2010), 48–57. https://doi.org/10.14778/1920841.1920853
[19] 2019. Apache Parquet. https://parquet.apache.org/
[20] OpenStreetMap contributors. 2019. US Northeast dump obtained from https://download.geofabrik.de/. https://www.openstreetmap.org
[21] NYC Taxi & Limousine Commission. 2020. TLC Trip Record Data. https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page.
[22] TPC. 2019. TPC-H. http://www.tpc.org/tpch/
[23] Evan Hallmark. 2020. Daily Historical Stock Prices (1970 - 2018). https://www.kaggle.com/ehallmar/daily-historical-stock-prices-1970-2018.

## 6. Use of AI Tools for Course Assignments, Theses, and Research Papers

Prof. Dr. Volker Markl
Chair of Database Systems and Information Management (DIMA)
Technische Universität Berlin

June 7, 2023

**Policy Statement.**

1. Effective immediately, as a general policy, we do not permit the use of ChatGPT or other large language models (LLMs) in DIMA for the preparation of research papers, technical reports, theses, PhD dissertations, or any other original work.

2. The use of LLMs shall be strictly prohibited, as it on the one hand might prevent a fair grading of student performance, and on the other hand present the very serious risk of plagiarism. Similar to copying code from *Stack Overflow*, there is a spectrum with respect to when plagiarism occurs. However, in the scientific world one should be particularly aware and be careful with respect to this.

3. Every DIMA member should be aware of this policy and also point this out to your students and mentees. For course/project reports and theses, we will require students to sign a form that they confirm that they did not use any LLM tools, such as ChatGPT for the writing of their theses. Of course, this also applies to research papers and PhD dissertations.

4. Please be mindful of this matter and bring any violations of this policy to my attention, so that DIMA will be an environment free of the risk of plagiarism.

**Please sign below to acknowledge that you have read this policy and will adhere to it.**

**Signature:** Riccardo Marin **Date: 20.09.2023**