Research Statement | Laurent Bindschaedler

Building the next generation of massive-scale data management and AI systems at the intersection of operating systems, databases, and machine learning.

The rapid advancement of artificial intelligence, particularly large language models (LLMs) and complex AI applications, has fundamentally transformed the landscape of data management and computing systems. As AI workloads become increasingly sophisticated and agentic, traditional single-model approaches reveal significant limitations: restricted context windows, insufficient parallelism, high inference costs, and challenges in maintaining reproducibility at scale. Meanwhile, the explosion of graph-structured data, multi-modal datasets, and decentralized applications continues to strain existing infrastructure. Organizations now face a dual challenge: building systems that can efficiently support complex AI workloads while managing ever-growing datasets across distributed environments.

My research addresses these challenges by building next-generation systems for massive-scale data management and AI at the intersection of operating systems, databases, and machine learning. I lead the Data Systems Group (DSG) at the Max Planck Institute for Software Systems, where we design and build systems that overcome the current limitations in AI and data applications. Our work spans complex AI systems (integrated environments where multiple models and non-model components work together efficiently), graph analytics, learned database components, blockchain systems, and multi-modal data processing. We combine machine learning techniques with principled systems engineering to create solutions that are not just powerful but also robust, efficient, and adaptable to the demands of modern applications. This work builds upon foundational research on load balance and parallelism in distributed systems that I developed during my doctoral studies at EPFL and postdoctoral research at MIT CSAIL, where I addressed challenges like data skew, stragglers, and resource contention that continue to inform how we approach efficiency and scalability in today's AI systems.

My research is experimental: it involves building systems and evaluating them. I draw much inspiration from industry, talks, meetups, personal interaction with other researchers, and social media. Most of my research so far has focused on challenges that real systems face, directly gathered from the research community and practitioners. The industry experience that I acquired before my graduate studies, mainly through my startup, also weighs heavily in my approach to problem selection. I take a principled, systems-oriented approach to understanding and solving problems: measuring and understanding limitations of existing systems, building quick prototypes to validate observations, and finally designing new abstractions to generalize the solution. I am also a firm believer in open-source and open research and make all my research artifacts readily available to researchers and practitioners, particularly all source code[1]. Finally, I aim to make scientific research and contributions more accessible to the general public, which I have done so far through blog posts[2] and science communication movies[3].

The rest of this statement describes the research directions I am pursuing with my group and collaborators, and concludes with the foundational work that inspired them.

Complex AI Systems

Current AI applications predominantly rely on single models like GPT or Llama. However, as the demand for more complex and sophisticated AI applications grows, thin-wrapper applications around cloud-based models reveal significant limitations. In parallel, as models keep growing, they become more computation and power-hungry, introducing significant systems challenges around efficiency and resource management for large-scale deployments. My group designs and builds components and end-to-end systems that overcome these limitations, creating AI systems that are not just powerful but also robust, efficient, and adaptable to the demands of modern applications.

Cache Saver: Efficient and Reproducible LLM Inference [Paper] [Code]

Problem LLM inference is expensive, both financially and environmentally. Organizations running LLM-based applications face significant costs from repeated API calls, while the carbon footprint of large-scale inference continues to grow. Moreover, the non-deterministic nature of LLM outputs poses challenges for reproducibility in research and production settings.

Solution and Impact We developed Cache Saver, a modular, plug-and-play framework for high-level LLM inference optimizations [13]. Cache Saver uses a namespace-aware list-valued cache to ensure statistical integrity and reproducibility across experiments. The framework reduces inference costs by approximately 25% and CO2 emissions by

[1] https://github.com/bindscha
[2] https://binds.ch/blog
[3] https://www.imdb.com/name/nm9978952

approximately 35% on average, while maintaining the statistical properties required for valid experimental results. Cache Saver has been accepted at EMNLP 2025 and is available as open-source software.

Code Intelligence [SQL-of-Thought] [LoRACode]

Problem Leveraging LLMs for code-related tasks presents unique challenges. Text-to-SQL systems must handle complex schema linking and error correction, while code embedding models require extensive fine-tuning that is computationally prohibitive.

Solution and Impact We developed two complementary approaches. SQL-of-Thought introduces a multi-agent framework for text-to-SQL that decomposes the task into schema linking, subproblem identification, query plan generation, SQL generation, and a guided correction loop [9]. Unlike prior systems that rely only on execution-based static correction, we introduce taxonomy-guided dynamic error modification informed by in-context learning. SQL-of-Thought achieves state-of-the-art results on the Spider dataset and its variants. LoRACode addresses the fine-tuning challenge through parameter-efficient adaptation for code embeddings, reducing trainable parameters to under 2% while maintaining competitive performance [8]. SQL-of-Thought has been accepted at DL4C@NeurIPS 2025, and LoRACode at DL4C@ICLR 2025.

LLM-Enhanced Databases [Paper]

Problem Integrating LLMs into database queries promises powerful semantic analysis capabilities, but existing approaches using general-purpose LLMs are prohibitively expensive for processing millions of rows in OLAP workloads.

Solution and Impact We developed IOLM-DB, which makes LLM-enhanced database queries practical through query-specific model optimization [12]. Rather than using expensive general-purpose LLMs for all queries, we create specialized lightweight models tailored to specific analytical tasks, enabling scalable processing of millions of rows at a fraction of the cost. This work has been accepted at DOLAP 2025.

Agentic AI Infrastructure

A key initiative in my group focuses on developing agentic AI infrastructure—end-to-end environments where multiple models and non-model components are integrated to work together efficiently. As AI workloads become more sophisticated, traditional single-model approaches reveal limitations such as restricted context windows and insufficient parallelism. Our research targets optimizing agentic workloads through advanced scheduling, caching, and resource management techniques. We are exploring how to make using thousands of LoRA adapters practical, and addressing the fundamental limitations of large language models through techniques like efficient updatable state management.

Graph and Learned Systems

Graph-structured data and machine learning are increasingly intertwined: graph neural networks (GNNs) have become essential for learning on relational data, while learned components promise to revolutionize traditional database systems. My group addresses the systems challenges at this intersection, from scaling GNN training to building updateable learned indexes.

Triskelion: Scaling Large Graph Neural Network Training

Problem Graph Neural Networks (GNNs) have become the de facto models for machine learning on graph datasets. As the size of real-world graph datasets and the complexity of GNN architectures increase, there is a growing need for efficient solutions to train on large graphs with billions of edges. Unfortunately, existing GNN training systems struggle to scale to large datasets due to the high overhead of exchanging data between graph partitions during their sampling phase.

Solution and Impact My group developed Triskelion, a GNN training framework designed to handle large graphs efficiently. Triskelion eliminates all remote accesses between partitions during sampling, thus avoiding the need for expensive intermediate data exchange. To counteract the potential loss in model accuracy resulting from its isolated sampling approach, Triskelion actively repartitions the graph during training and dynamically adjusts the weight of gradients during aggregation. This strategy allows each node to be adequately exposed to the entire graph while countering sampling bias, thereby maintaining the model's accuracy. We demonstrated that Triskelion trains GNNs up to 13$\times$ faster than state-of-the-art systems while maintaining comparable accuracy and can handle graphs with trillions of edges on commodity hardware, a new milestone in GNN training capacity.

GNN Partitioning Strategies [Paper]

Complementing our work on Triskelion, we have conducted a comprehensive survey on graph neural network partitioning strategies [15]. This work systematically analyzes the trade-offs between different partitioning approaches and their impact on GNN training performance, providing guidance for practitioners and researchers working with large-scale graph learning systems. This survey has been accepted at GRADES-NDA 2025.

Updateable Learned Indexes

Learned indexes promise significant performance gains over traditional B-trees by using machine learning models to predict key positions. However, existing learned index structures struggle with updates, limiting their applicability in dynamic workloads. We are developing concurrently updateable learned index structures that maintain the performance benefits of learned models while supporting efficient insertions and deletions. This work addresses a fundamental limitation that has prevented wider adoption of learned indexes in production database systems.

Unified Bespoke Graph Processing

Efficiently handling different graph workloads within the same system is challenging due to the varying characteristics of graph applications. For instance, analytics and graph mining applications rely on sorted adjacency lists incompatible with graph queries and transactions. Graph partitioning across machines is also tricky. As a result, most graph processing systems target specific workloads and employ specialized data structures, requiring costly pre-processing steps that can take longer than the actual computation [11].

In collaboration with Prof. Ashvin Goel's group at the University of Toronto, we are building a graph processing system that can efficiently execute different applications by automatically fine-tuning configuration and synthesizing optimized components. Our first focus was designing a graph store that can support efficient updates at arbitrary granularity while processing long analytics queries. We are now expanding our store design to support a broader range of applications and distributed storage and execution. Finally, we plan to support mining anomalies and insights from streaming graph data.

Benchmarking Learned Systems [Paper] [Slides] [Video]

The use of machine learning to tune data management systems or synthesize components tailored to a specific problem instance has recently become a popular research direction. The promise of learned systems is their ability to automatically adapt to new workloads, data, or hardware without time-consuming tuning by humans, thereby dramatically reducing the cost and accessibility of data analytics. However, traditional benchmarks such as TPC or YCSB that evaluate performance under a stable workload and data distribution are insufficient to characterize these systems due to the latter's ability to overfit to the benchmark.

I have presented several ideas for designing new benchmarks that are better suited to evaluate learned systems [3]. New benchmarks should abstain from using fixed workloads and data distributions as their characteristics are easy to learn. Similarly, they should strive to measure adaptability through descriptive statistics and outliers rather than average metrics that "hide" too much information. I also proposed techniques and metrics to incorporate model training and cost savings into benchmark results, two key characteristics that can no longer be ignored in learned systems.

Blockchain and Decentralized Systems

Blockchains and decentralized applications present unique systems challenges: they require efficient analytics over append-only ledgers, trustless query execution, and careful requirements engineering for governance mechanisms. My group addresses these challenges through dedicated blockchain infrastructure and principled approaches to decentralized systems design.

AlterEgo: A Dedicated Blockchain Node For Analytics [Paper]

Blockchains today amass terabytes of transaction data that demand efficient and insightful real-time analytics for applications such as smart contract hack detection, price arbitrage on decentralized exchanges, or trending token analysis. Conventional blockchain nodes, constrained by their RPC APIs, and specialized ETL-based blockchain analytics systems grapple with a trade-off between materializing pre-calculated query results and analytical expressiveness.

My group developed AlterEgo [10], a blockchain node architected specifically for analytics that maintains parity with traditional nodes in ingesting consensus-produced blocks while integrating a robust analytics API. Our prototype supports efficient transactional and analytical processing while circumventing the rigidity of ETL workflows, offering a better trust model, enabling distributed and collaborative querying, and achieving significant performance improvements over the state-of-the-art. We are currently expanding AlterEgo's capabilities by integrating sophisticated anomaly detection algorithms and machine learning models and incorporating support for decentralized, trustless query execution.

Decentralized Systems Engineering [DAO Workflows] [Prediction Markets]

Building secure, incentive-aligned decentralized platforms requires rigorous requirements engineering—a discipline that has been underexplored in the Web3 space. We have developed approaches to address this gap through two complementary projects. First, we proposed a computational decision support workflow for requirements engineering in Decentralized Autonomous Organizations (DAOs), addressing the unique governance challenges these organizations face [6]. Second, we conducted a requirements analysis for a decentralized mathematics prediction

market, demonstrating how requirements-driven design can be applied to novel decentralized applications [7]. Both works have been accepted at RE4Web3 2025, setting new standards for early-stage requirements engineering in decentralized systems.

Multi-Modal Data Systems

SkyPulse: Multi-Modal Satellite Data Augmentation

Satellite imagery has become a critical source of information for businesses and government agencies. Applications include monitoring essential infrastructure or disaster-affected areas, detecting weather patterns, and tracking traffic patterns. As this technology expands, data augmentation, integrating and supplementing satellite data with multi-modal data sources, has become imperative to provide a comprehensive view of monitored areas. This need is emphasized by emerging applications such as machine learning, which require diverse, high-quality data. Consequently, robust storage and serving solutions are needed to handle the data influx and support data augmentation processes.

This project aims to design an efficient end-to-end system for satellite data augmentation with multi-modal data in collaboration with the Swiss Federal Office for Defence Procurement (Armasuisse). We built SkyPulse[4], a distributed system that uniquely integrates satellite data with varied sources ranging from social media to open-source intelligence. This system addresses the growing need for multi-modal data analysis and high-quality datasets in machine learning applications. SkyPulse provides efficient ingestion and storage for data sources, advanced data fusion capabilities, high-performance data export and serving for third-party analytics systems such as Spark or machine learning training. We showcased our system through example applications and datasets, including tracking natural disasters and combining weather observation cameras with satellite imagery for climate change studies. We have recently been working on an extension of this work to support real-time critical infrastructure monitoring (e.g., nuclear power plants) through augmented satellite data and other data sources.

Foundational Work: Mitigating Load Imbalance in Cluster Applications

Load imbalance in distributed applications refers to situations where different machines take different amounts of time to finish their assigned tasks, wasting resources and limiting parallelism as the other machines remain idle. Load imbalance takes various forms and has many possible causes, including skewed data partitioning, variance in the amount of generated intermediate state, data-dependent processing times or filtering, irregular memory accesses, hardware heterogeneity or failures, and interference from background tasks. Therefore, improving load balance is crucial for application developers and cluster operators as more balanced systems generally benefit from higher performance, faster job completion times, and better overall resource utilization.

My thesis proposed *Scatter computing*, a novel architecture for distributed data management and analytics systems that mitigates the performance impact of load imbalance [1]. Intuitively, addressing load imbalance requires decoupling tasks from the machines processing them, allowing the system to pool resources and respond to changing load conditions. The Scatter architecture entails rethinking distributed storage abstractions to avoid storage hotspots and designing coordination-avoidance techniques to enable multiple machines to share the work of processing a single task with low synchronization overhead. As a result, Scatter systems are highly resilient to imbalanced situations and often perform better than traditional systems in uniform cases.

I used the Scatter architecture to address load balance in a diverse set of applications, including graph analytics, general-purpose analytics, distributed databases, and machine learning, as described in the following sections.

Scale-out Graph Processing from Secondary Storage [Paper] [Slides] [Video] [Code]

Problem The availability of graph-structured data in domains ranging from social networks to national security has created renewed interest in designing systems to mine valuable information from such graphs. A serious impediment to this effort is that real-world graphs have skewed vertex degree distributions, and many graph algorithms exhibit irregular access patterns, making it hard to partition data and work. Moreover, the fast growth of graphs exacerbates these difficulties as datasets may not fit in the aggregate memory of all machines in the system.

Solution and Impact I designed Chaos, a scale-out graph processing system for secondary storage that enables analytics on very large graphs with trillions of edges [14]. Following the Scatter architecture principles, Chaos does away with elaborate partitioning schemes and instead spreads the data for each partition uniformly at random across all machines. We achieve high performance and load balance by leveraging this uniform data placement and a work-stealing strategy that allows multiple machines to process a single partition. As a result, Chaos is faster than competing systems by over an order of magnitude on many graph workloads and datasets. Chaos also pushed the limit of the graph size that can be processed in a cluster of commodity servers by analyzing the connectivity of a graph with over 250 trillion

[4] https://skypulse.mpi-sws.org

edges on 20 machines in a little over 10 hours, a feat previously only achieved by supercomputers. Five years later, Chaos still holds the 5th position in the Graph500 ranking of the largest graphs processed by computer.

Taming Skew in Large Scale Analytics [Paper] [Slides] [Video] [Code]

Problem Application runtimes in distributed data analytics frameworks such as Apache Hadoop and Spark are often unpredictable and underperforming on many input datasets and deployments due to, e.g., data and compute skew, slow or faulty machines, cluster heterogeneity. This load imbalance introduces stragglers that cause other machines to sit idle, degrading performance for the entire parallel job. The underlying cause for these issues is static work partitioning, i.e., creating fixed-size tasks that cannot be dynamically broken down. Since these systems inherently require that a single worker process each task, they have little recourse but wait for stragglers.

Solution and Impact I built Hurricane, a high-performance general-purpose analytics system inspired by the Scatter architecture that generalizes the work-stealing strategy of Chaos to design an adaptive task partitioning scheme that achieves fast execution times and high cluster utilization [5]. Hurricane introduced a new "data bag" storage abstraction and a task cloning strategy that allows workers to share a task and automatically adapts parallelism at runtime. Therefore, Hurricane provides orders-of-magnitude performance improvements in skewed workloads without introducing performance degradation in uniform workloads and datasets. After the paper's publication, I worked with a team of students from the University of Toronto to implement a fully productized open-source version of Hurricane.

Disaggregation for Distributed LSM-based Databases [Paper] [Slides] [Video] [Code]

Problem Distributed databases that use Log-Structured Merge-Tree (LSM) storage engines such as RocksDB often suffer from unpredictable performance and low utilization due to skew and background operations. Skew causes CPU and I/O imbalance, which degrades overall throughput and response time. Current LSM-based databases address skew by resharding data across machines. However, this operation is expensive because it involves bulk data migration, which affects throughput and response time for users. Similarly, background operations such as flushing and compaction can cause significant I/O and CPU bursts, leading to severe latency spikes, especially for queries spanning multiple machines such as range queries or transactions. These problems are hard to address in existing systems because the storage engines operate independently of each other and thus are unaware of resource usage and background operations on different machines.

Solution and Impact I designed Hailstorm, an LSM-optimized distributed file system that runs transparently below database deployments and reduces storage contention while providing cluster awareness to LSM storage engines [2]. Hailstorm automatically rebalances data from each storage engine across machines, reducing storage contention and enabling overloaded machines to offload background tasks. The Hailstorm design demonstrates the benefits of my Scatter architecture in user-facing systems. Using Hailstorm, I doubled the throughput of the widely popular MongoDB database in write-intensive workloads and multiplied scan throughput by over 20×. Hailstorm also achieves over 2× faster throughput for PingCAP's TiDB on industry-standard benchmarks TPC-C and TPC-E.

Alleviating Degradation With Non-IID Data in Machine Learning

Problem Stochastic Gradient Descent (SGD) is a popular optimization algorithm that is used by a variety of machine learning applications due to its fast learning rate and small memory footprint. However, SGD is notoriously hard to parallelize in traditional MapReduce environments due to its sequential nature and high communication overheads. Moreover, most parallel implementations of SGD, especially Apache Spark and Parameter Server, assume independent identically distributed data (IID) across partitions. Unfortunately, this assumption does not always hold in practice, especially in the presence of skewed samples or when the partitioning across workers itself exhibits skew. In such cases, the SGD algorithm may suffer from significant performance degradation or fail to converge.

Solution and Impact I built Snowball, a distributed load-balanced implementation of SGD. Snowball supports asynchronous gradient updates through a distributed key-value store and uses a data bag abstraction to allow parallel workers to sample from the entire dataset without partitioning it, reducing the impact of skew. Snowball leverages the Scatter architecture to mitigate a different type of skew in a seemingly perfectly load-balanced situation: all workers always perform exactly the same amount of work, but some workers perform unnecessary or counterproductive work. As a result, Snowball achieves high compute and storage utilization on all machines in the cluster while ensuring faster convergence even in the presence of significant skew. Snowball's shared data bag design also dampens the effects of asynchronous weight updates that often impact Parameter Server implementations.

Real-Time Graph Pattern Mining [Paper] [Slides] [Video] [Code]

Extracting insight from streams of data is becoming a must-have feature in many systems that support business decisions. Indeed, periodic recomputation on snapshots of the data is wasteful and often too slow to support interactive data analysis. I explored real-time analytics in the context of graphs.

Problem Graph pattern mining, i.e., finding instances of interesting patterns (matches) in a graph dataset, has wide-ranging applications in social networks, chemistry, credit card fraud detection, and semantic web. Since graph mining applications can easily take hours to run, even on modest-sized graphs, it is desirable to incrementally maintain

the set of all matches as the graph is updated instead of recomputing everything from scratch upon receiving updates. Mining dynamic graphs introduces a new set of challenges. First, new algorithms are necessary to ensure correctness under updates and not miss any matches or produce duplicates. Second, a different software architecture is required to support high-throughput streams with thousands to millions of updates per second while maintaining mining results in real-time. In particular, this architecture must assign updates to workers in a balanced fashion while minimizing data transfer and synchronization across workers at scale.

Solution and Impact I designed Tesseract, a distributed graph mining system that automatically executes any existing static algorithm in dynamic graphs [4]. Tesseract introduces a novel change detection algorithm that efficiently determines the exact modifications for each update. Moreover, by favoring task parallelism over data parallelism, Tesseract can decompose a stream of graph updates into per-update mining tasks and dynamically assign these tasks to a set of distributed workers. Tesseract supports millions of updates per second with low latency and is several orders-of-magnitude faster than periodic recomputation on snapshots. Finally, Tesseract outperforms static mining systems on the entire graph, despite the overheads of supporting graph updates, thanks to its low memory requirements, efficient storage, and communication.

Bibliography

[1] Laurent Bindschaedler. 2020. *An Architecture for Load Balance in Computer Cluster Applications*. Technical Report. EPFL.

[2] Laurent Bindschaedler, Ashvin Goel, and Willy Zwaenepoel. 2020. Hailstorm: Disaggregated Compute and Storage for Distributed LSM-based Databases. In *Proceedings of the 25th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*. ACM.

[3] Laurent Bindschaedler, Andreas Kipf, Tim Kraska, Ryan Marcus, and Umar Farooq Minhas. 2021. Towards a Benchmark for Learned Systems. In *37th IEEE International Conference on Data Engineering Workshops, ICDE Workshops 2021, Chania, Greece, April 19-22, 2021*. IEEE, 127–133. `https://doi.org/10.1109/ICDEW53142.2021.00029`

[4] Laurent Bindschaedler, Jasmina Malicevic, Baptiste Lepers, Ashvin Goel, and Willy Zwaenepoel. 2021. Tesseract: Distributed, General Graph Pattern Mining on Evolving Graphs. In *Proceedings of the 16th EuroSys Conference (EuroSys '21)*. ACM.

[5] Laurent Bindschaedler, Jasmina Malicevic, Nicolas Schiper, Ashvin Goel, and Willy Zwaenepoel. 2018. Rock You Like a Hurricane: Taming Skew in Large Scale Analytics. In *Proceedings of the Thirteenth EuroSys Conference*. ACM, 20.

[6] Quentin Botha, Laurent Bindschaedler, and Christoph Siebenbrunner. 2025. A Computational Decision Support Workflow for Requirement Engineering in DAOs. In *Proceedings of the 1st International Workshop on Requirements Engineering for Web3 (RE4Web3)*.

[7] Quentin Botha, Laurent Bindschaedler, and Christoph Siebenbrunner. 2025. A Requirements Analysis for a Decentralized Mathematics Prediction Market. In *Proceedings of the 1st International Workshop on Requirements Engineering for Web3 (RE4Web3)*.

[8] Saumya Chaturvedi, Aman Chadha, and Laurent Bindschaedler. 2025. LoRACode: LoRA Adapters for Code Embeddings. In *Deep Learning for Code Workshop at ICLR 2025*.

[9] Saumya Chaturvedi, Aman Chadha, and Laurent Bindschaedler. 2025. SQL-of-Thought: Multi-agentic Text-to-SQL with Guided Error Correction. In *Deep Learning for Code Workshop at NeurIPS 2025*.

[10] Qi Guo, Mahdi Alizadeh, Ali Falahati, and Laurent Bindschaedler. 2024. AlterEgo: A Dedicated Blockchain Node For Analytics. In *Proceedings of the 7th International Workshop on Edge Systems, Analytics and Networking*. 7–12.

[11] Jasmina Malicevic, Baptiste Lepers, and Willy Zwaenepoel. 2017. Everything you always wanted to know about multicore graph processing but were afraid to ask. In *2017 {USENIX} Annual Technical Conference ({USENIX}{ATC} 17)*. 631–643.

[12] Bardia Mohammadi and Laurent Bindschaedler. 2025. The Case for Instance-Optimized LLMs in OLAP Databases. In *Proceedings of the 27th International Workshop on Design, Optimization, Languages and Analytical Processing of Big Data (DOLAP)*.

[13] Nearchos Potamitis, Lars Henning Klein, Bardia Mohammadi, Chongyang Xu, Attreyee Mukherjee, Niket Tandon, Laurent Bindschaedler, and Akhil Arora. 2025. Cache Saver: A Modular Framework for Efficient, Affordable, and Reproducible LLM Inference. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

[14] Amitabha Roy, Laurent Bindschaedler, Jasmina Malicevic, and Willy Zwaenepoel. 2015. Chaos: Scale-out Graph Processing from Secondary Storage. In *Proceedings of the 25th Symposium on Operating Systems Principles*. ACM, 410–424.

[15] Chongyang Xu and Laurent Bindschaedler. 2025. Everything You Wanted to Know About Graph Neural Network Partitioning (But Were Afraid to Ask). In *Proceedings of the 8th Joint Workshop on Graph Data Management Experiences & Systems (GRADES) and Network Data Analytics (NDA)*.