

# CS848 Fall 2026: Indexing for Modern Data Processing

## Introduction

Xiao Hu

September 10, 2026

# Course Information

---

- Time: 10 AM – 11:20 AM every Tuesday and Thursday
- In-person at DC 2568
- Email: [xiaohu@uwaterloo.ca](mailto:xiaohu@uwaterloo.ca)
  - Emails about this class – please include **[CS848 F26]** in subject line
- Office hour by appointment
- Resources:
  - LEARN course site (should appear in your account)
  - Course website <https://cs.uwaterloo.ca/~xiaohu/courses/CS848/CS848-F26.html>

# About Me

---

- Xiao Hu
  - Assistant Professor at School of Computer Science
  - <https://cs.uwaterloo.ca/~xiaohu/>
- Education
  - PhD from HKUST
  - BS from Tsinghua University
- Experiences
  - Postdoc at Duke University
  - Visiting Researcher at Algorithm Group of Google Research
  - Visiting Researcher at Simons Institute for the Theory of Computing
  - Visiting Researcher at EECS Berkeley

# My Research Interest

---

- Database theory and its applications to data systems
- What is database theory?
  - A bridge between theoretical computer science and practical data management.
  - It studies principles, algorithms, and models for designing, optimizing, and querying data systems.
- My view of indexing
  - An **algorithmic preprocessing problem** for efficient data retrieval
  - Given data and a query workload, construct an auxiliary representation that reduces future query costs
  - Balance **construction, space, query, update, and accuracy** costs

# Why this course?

---

## ■ Why indexing now?

- Vector search has become more and more important (semantic search, recommendation, RAG, and multimodal retrieval)
- Real applications require hybrid search (similarity + filters + joins)
- The research frontier is moving very rapidly (many open questions in accuracy, latency, memory, updates, and optimization)

<https://vldb.org/2026/program.html>

- Students need more than library-level knowledge

## ■ What is the ultimate goal?

- Understand the capabilities, principles, limitations, and applicability of different indexing techniques.

# What you can learn from this course?

---

- Understand modern indexing
  - Explain the core ideas behind relational, vector, and hybrid indexes.
  - Recognize common principles across different index families.
- Evaluate indexing techniques
  - Analyze construction, space, query, and update costs.
  - Evaluate accuracy, recall, latency, and scalability.
  - Connect theoretical guarantees with empirical performance.
- Conduct research on indexing
  - Identify a meaningful research problem.
  - Build, benchmark, or extend an index on real data.
  - Design reproducible experiments and communicate the results.

# Introduce Yourself

---

- Your background
  - Name
  - School and program
  - Year in the program
  - Your academic advisor
- Research Interest
  - Current research directions
  - One past or ongoing project you would like to share
- Experience with using index (No prior experience is totally fine!)
- Goals of this course
  - What do you want to learn?

# What is a Database Management System?

A DBMS manages the storage, organization, access, and processing of data.

## Store

Represent data,  
Place data across  
memory, SSD,  
and distributed  
storage.

## Organize

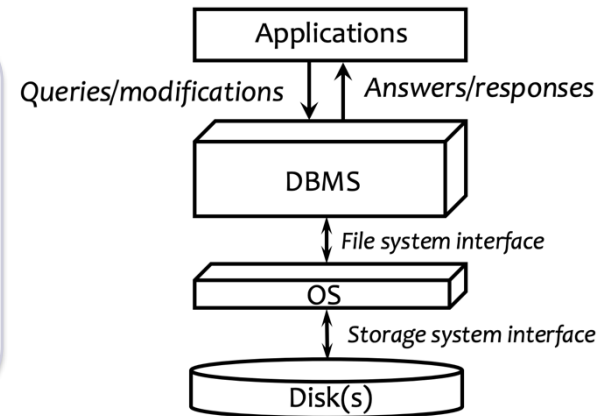
Maintain layouts,  
partitions,  
statistics, and  
**indexes.**

## Query

Translate  
information  
needs into  
executable plans.

## Serve

Support updates,  
concurrency,  
reliability, and  
access control.



**Indexes connect the physical layout of data with efficient query processing.**

# Modern Queries in Disguise

“Find 10 recent database papers most relevant to given paper.”

IEEE TRANSACTIONS ON JOURNAL NAME, MANUSCRIPT ID

1

## Efficient and robust approximate nearest neighbor search using Hierarchical Navigable Small World graphs

Yu. A. Malkov, D. A. Yashunin

**Abstract** — We present a new approach for the approximate K-nearest neighbor search based on navigable small world graphs with controllable hierarchy (Hierarchical NSW, HNSW). The proposed solution is fully graph-based, without any need for additional search structures, which are typically used at the coarse search stage of the most proximity graph techniques. Hierarchical NSW incrementally builds a multi-layer structure consisting from hierarchical set of proximity graphs (layers) for nested subsets of the stored elements. The maximum layer in which an element is present is selected randomly with an exponentially decaying probability distribution. This allows producing graphs similar to the previously studied Navigable Small World (NSW) structures while additionally having the links separated by their characteristic distance scales. Starting search from the upper layer together with utilizing the scale separation boosts the performance compared to NSW and allows a logarithmic complexity scaling. Additional employment of a heuristic for selecting proximity graph neighbors significantly increases performance at high recall and in case of highly clustered data. Performance evaluation has demonstrated that the proposed general metric space search index is able to strongly outperform previous opensource state-of-the-art vector-only approaches. Similarity of the algorithm to the skip list structure allows straightforward balanced distributed implementation.

**Index Terms** — Graph and tree search strategies, Artificial Intelligence, Information Search and Retrieval, Information Storage and Retrieval, Information Technology and Systems, Search process, Graphs and networks, Data Structures, Nearest neighbor search, Big data, Approximate search, Similarity search

### 1 INTRODUCTION

Constantly growing amount of the available information resources has led to high demand in scalable

errors. The quality of an inexact search (the recall) is defined as the ratio between the number of found true near-

# Modern Queries in Disguise

“Find 10 recent database papers most relevant to given paper.”

## Structured filter

year  $\geq 2024$   
venue  $\in \{\text{SIGMOD, VLDB, ICDE}\}$

Structural Filtered Conditions

## Vector similarity

Embed the query papers; Compare it with all other paper embeddings

Approximate relevance

## Top-k result

Combine constraints + scores and return the most relevant papers.

Filtered ANN Query

| paper_id | year | venue | keywords | ... |
|----------|------|-------|----------|-----|
|          |      |       |          |     |
|          |      |       |          |     |
|          |      |       |          |     |
|          |      |       |          |     |

### Implementing Semantic Join Operators Efficiently

Immanuel Trummer  
Cornell University  
Ithaca, USA  
itrummer@cornell.edu

Download

#### ABSTRACT

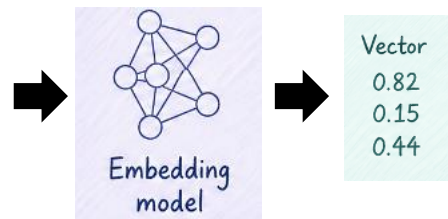
Semantic query processing engines often support semantic joins, enabling users to match rows that satisfy conditions specified in natural language. Such join conditions can be evaluated using large language models (LLMs) that solve novel tasks without task-specific training.

Currently, many semantic query processing engines implement semantic joins via nested loops, involving the LLM to evaluate the join condition on row pairs. Instead, this paper proposes a novel algorithm, inspired by the block nested loop join operator implementation in traditional database systems. The proposed algorithm integrates batches of rows from both input tables into a single prompt. The goal of the LLM invocation is to identify all matching row pairs in the current input. The paper introduces formulas that can be used to optimize the size of the row batches, taking into account constraints on the size of the LLM context window (limiting both input and output size). An adaptive variant of the proposed algorithm refers to cases in which the size of the output is difficult to estimate. A formal analysis of asymptotic processing costs, as well as empirical results, demonstrates that the proposed approach reduces costs significantly and performs well compared to join implementations used by recent semantic query processing engines.

This paper focuses on a semantic version of a classical relational operator: the relational join. Semantic joins, as defined by systems like LITVUS [25], enable users to define the join condition in natural language. Note that this paper does not explicitly focus on equality joins. Instead, it focuses on general theta-joins [26] with natural language predicates. Such join operators are useful in the following example scenarios.

Example 1.1 To investigate a large corporation's procurement plan to analyze a large collection of emails. The goal is to compare emails to statements made by executives of that company. Of particular interest are instances where an email contradicts statements made by the defendants. This can be modeled as a join between two data sets, one containing statements and the other one containing emails. The join predicate can be formulated in natural language as "pairs of documents that contradict each other".

Example 1.2 A Website enables users to enter ads for used goods as free text, as well as free text descriptions of items they are searching for. Free text descriptions may contain detailed descriptions of various aspects of the items offered (or desired). For instance, a user looking for a new table may include constraints and preferences with regard to material, color, size, or state (e.g., "no coffee stains"). The Website wants to introduce a feature that supports users in



# New Data, New Queries, New Search Challenges

## Data

Volume and velocity  
**High-dimensional** vectors  
Unstructured and multimodal content

## Queries

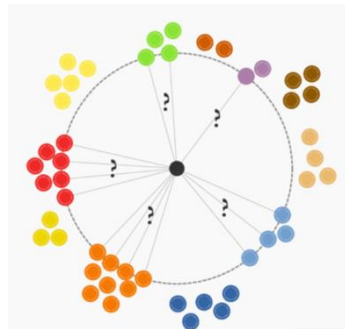
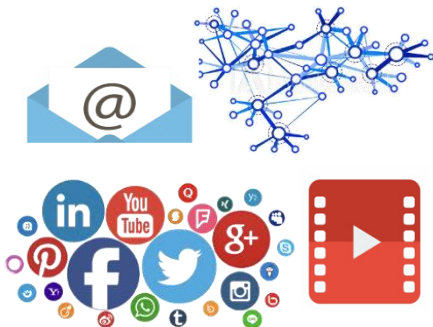
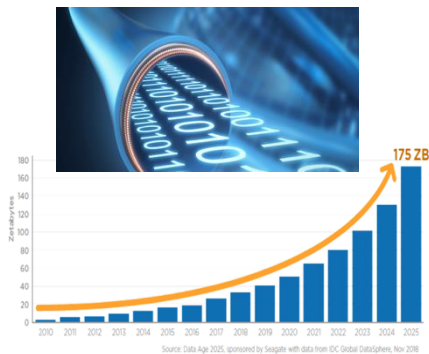
Exact predicates and range conditions  
Approximate similarity and top-k search  
Semantic and hybrid retrieval

## Systems

Memory, SSD, and distributed storage  
GPUs and specialized hardware

## Guarantees

Correctness vs. approximation (recall)  
Latency and throughput  
Space, construction, and update cost



# The Data-System Landscape Is Expanding

Modern systems increasingly support more than one data model and more than one notion of search.

## Relational systems

Tables and attributes  
Exact lookup and ranges  
Joins and aggregation

Typical access paths:  
B-trees, hashing, filters

## Vector systems

Embedding vectors  
Nearest-neighbor search  
Approximate relevance

Typical access paths:  
LSH, quantization, graphs

## Hybrid systems

Vectors plus metadata  
Filters, keywords, and equi-joins  
Semantic operators

Open question:  
How should these interact?



ORACLE



# One Query, Three Indexing Worlds

---

## Relational index

Organize ordered keys, hashes, or sketches

Question:  
Which tuples satisfy the predicate?

## Vector index

Partition, compress, hash, or connect points in vector space.

Question:  
Which objects are approximately nearest?

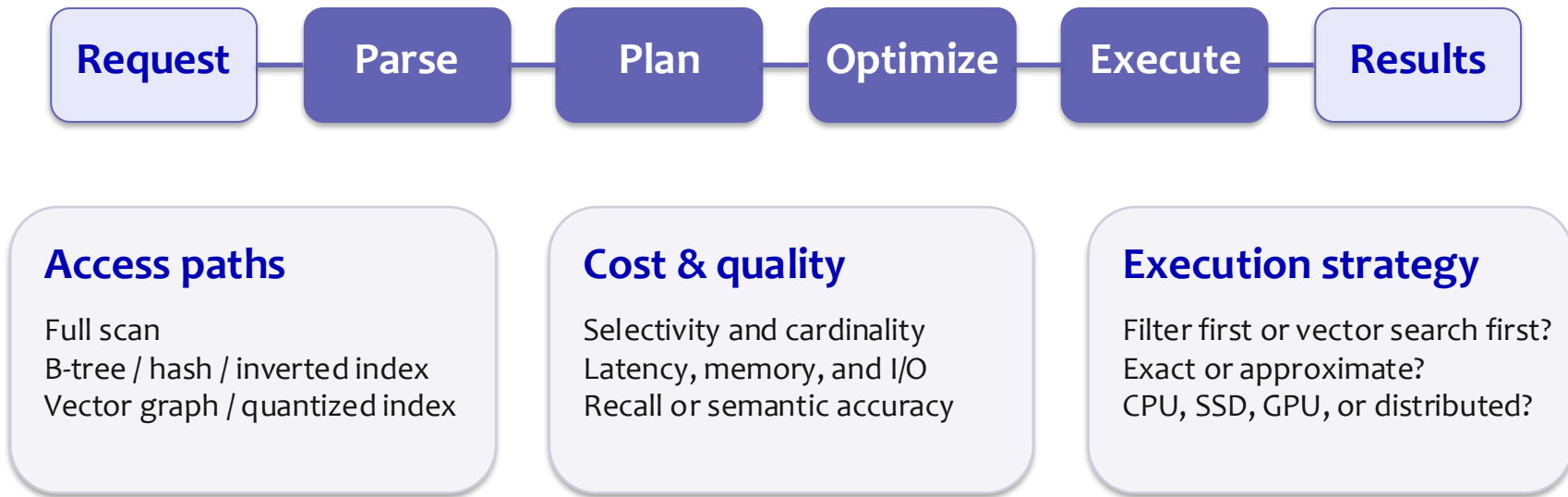
## Hybrid index

Coordinate metadata filtering with similarity search.

Question:  
Which execution order is best? Or, how to design a better hybrid index?

**Shared principle: preprocess data now to avoid unnecessary work later.**

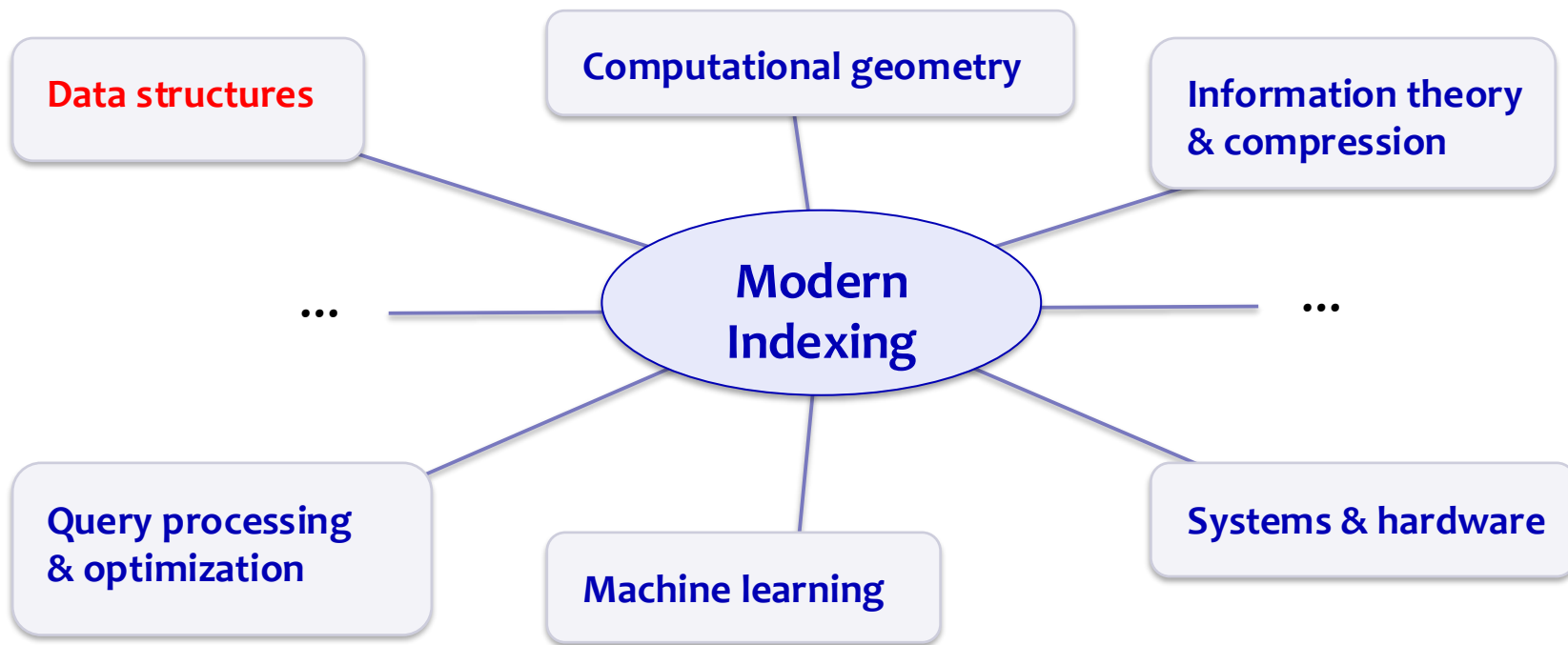
# How Are Modern Search Queries Processed?



**An index becomes useful only when the optimizer can choose and use it well.**

# A Holistic View of Indexing

---



# Scope of This Course

---

## Part I: Classical

B-trees and hashing  
Spatial indices  
Bloom and Cuckoo filters  
Sketches  
Learned indexes  
LSM trees

## Part II: Vector

Inverted index  
Locality-sensitive hashing  
Product quantization  
Graph-based ANN

## Part III: Hybrid

Filtered vector search  
Semantic query operators  
Multi-vector search  
Sparse-dense search

- Guest lectures will provide valuable industry perspectives (such as Alibaba, Google, Meta, Nvidia) on current research
- Keep in mind of the differences: exact vs. approximate; static vs. dynamic; in-memory vs. disk;

# Some background will help but not necessary

---

- Undergraduate-level databases (CS348)
- Algorithm design and analysis
  - Basic probability is helpful for randomized and approximate methods
- Programming skills for an implementation or benchmarking project
- Curiosity about algorithms, systems, and empirical evaluation

# Grading

---

- In-class participation (10%)
  - Important! Propose questions and answer questions
- Paper reviews (10%)
- Paper presentation (30%)
- Course project (50%)

# Paper Review Guide

---

- Reviews are submitted through LEARN
- Read the assigned papers before class
  - Starting from Week 4 (see submission details on LEARN)
- Reviews are due by 11:59 PM every Monday
- No late reviews are accepted
- Reviews must be your own work; AI-generated review content is not permitted

# Paper Review Organization Guide

---

## ■ Summary

- What research problem does this paper solve?
- How does this paper compare with prior works?
- What is the main contribution of this paper?
- How do the authors evaluate their proposed solution?

## ■ **Strengths:** at least three points

## ■ **Weaknesses:** at least three points

## ■ Detailed Comments:

**Think critically!**

- Expand the strengths and weaknesses, especially the weaknesses.
- **How does this work relate to your research?**
- Are there future directions inspired/enabled by this paper?

# Paper Presentation

---

- Each student presents one specific topic in one lecture
- One-hour presentation followed by 20 minutes of discussion
- Slides or notes should be uploaded to Learn two days before the presentation
- Bonus for early presentations (before reading break)
- The schedule may be adjusted based on enrollment numbers

# Paper Presentation Guide

---

- Background, Motivation and Formulation – **Use your own language**
  - Where, how and why does this problem arise?
  - What problem does this paper aim to solve?
  - How to formally formulate the problem? Such as definition, terminologies, notations (as simple as possible), and assumptions
- Review Previous Results
  - What are the main approaches used?
  - What are the best results (memory, recall, latency) achieved and what are the limitations?
- New Index and Algorithms – **A Running Example is strongly encouraged**
  - What is the high-level idea of the index?

# Paper Presentation Guide

---

- Step-by-step explanation of the algorithms (construction, query, update, etc)
- Summarize the technical points
- Analyze the complexity if there is any
- Practical Implementation
  - Implementations of the new index
  - Platform, workload, dataset, queries, baseline solutions
  - Experiment results and observations
- Discussion
  - Summarize the key contribution of the paper
  - Evaluate the strengths and weaknesses in your opinions
  - Open questions and future directions
  - Possible combination/application/inspiration for your own research

# Paper Presentation Evaluation

---

- It is strongly suggested to start two weeks earlier
- Be prepared with questions from the instructor and the audience
- If you don't understand some parts of the paper, please reach out to the instructor for discussion at least one week before your presentation
- It is suggested to send your slides or notes to the instructor for review, in case some important points are not covered

# Course Project Overview

---

- A term-long project completed individually or by at most two students
- Type 1: Build
  - Implement and tune an index; reproduce or extend published results
- Type 2: Benchmark
  - Compare index families using recall, latency, memory, update cost, etc.
- Type 3: Research
  - Explore a new algorithmic or systems question related to modern indexing
- All results should be reproducible and carefully documented

# Course Project Overview

---

- Proposal: Due 11:59 PM EST, Sunday, October 18
- Project presentations
  - Thursday, December 3 and Tuesday, December 8
  - All students are involved to grade the project presentations
- Final report: Due 11:59 PM EST, Friday, December 11
- Schedule office hours if you need to discuss project scope

# Examples of Course Projects

---

- Build and tune HNSW, IVF-PQ, or a learned index
- Benchmark index families on a real dataset
  - Measure recall, latency, memory, construction, and update costs
- Study filtered ANN under selective or correlated predicates
- Explore disk-based vector search
- Investigate sparse-dense vector search
- Investigate multi-vector vector search
- Design or optimize a semantic query operator
- **BE CREATIVE!**

# What do I expect from you

---

- **WORK HARD!**
- Start the course project early – it is impossible to finish a project in a week
- Start paper presentation early – it takes time to digest the paper
- Think critically when reviewing research papers
- Participate in discussions actively