Syllabus for the Course Data Structures & Databases

Format: Master Class + Digital Lab + AI-Augmented Learning

Duration: 15 academic weeks + exam
Classroom load: 23 hours of lectures (master class), 22 hours of laboratory classes
Independent work: 27 hours
Format: AI-Augmented Engineering Learning

1. General characteristics of the discipline

The discipline “Data Structures and Databases” is a mandatory component of the general scientific and professional module of the Master’s degree program “AI-Augmented Digital Systems Engineering” (specialization 09.04.02 “Information Systems and Technologies”).
The workload of the discipline is 2 ECTS credits (72 hours)
The intermediate certification form is an exam with an engineering defense.
The course is implemented in the 2nd semester and is a logical continuation of the course " Algorithms and Data Structures "in close connection with the end-to-end instrumental course"Algorithmization and Programming Languages".
The course provides a transition to:
from abstract data structures → to their physical implementation → to storage systems → to data architecture in AI systems.
The discipline uses the knowledge and skills developed in the course: "Fundamentals of AI and LLM: industrial and context engineering".

2. Place of the discipline in the program structure

The discipline serves as an engineering transition module between:

- algorithmic formalization,
- software engineering,
- architectural design,
- AI engineering and data processing systems.

While the first-semester course develops algorithmic thinking autonomy, this discipline enhances:

- an understanding of data storage principles;
- the ability to analyze query execution plans.
- skills in designing diagrams and indexes;
- understanding transactional consistency.
- ability to choose a storage model for a specific load.
- skills in measuring and analyzing productivity;
- fundamentals of designing a hybrid storage architecture for AI systems.

The course is the foundation for:

- disciplines on the architecture of information systems;
- ML and Data Engineeringmodules.
- distributed systems.
- project activity for 3-4 semesters.

3. Objectives of mastering the discipline

The objectives of mastering the discipline are:

- developing an engineering understanding of the physical organization of data storage;
- mastering indexing, transactional consistency, and query optimization mechanisms.
- development of experimental performance analysis skills;
- forming the ability to choose a storage model in the face of architectural compromises;
- preparation for designing data warehouses in AI-oriented systems;
- mastering the model of conscious and critical use of LLM in engineering analysis.

4. Objectives of the discipline

Within the framework of the discipline, the following tasks are solved:

- mastering the physical storage model (pages, indexes, logs);
- study of the relational model and optimization mechanisms;
- analysis of query execution plans.
- mastering the transactional model and isolation levels;
- comparison of SQL and NoSQL models;
- mastering the principles of vector search and embeddings storage;
- development of experimental hypothesis testing skills;
- development of engineering argumentation when choosing architectural solutions;
- creating a culture of working with LLM as an analytical tool.

5. Planned learning outcomes

As a result of mastering the discipline, the student must:

Know:

- principles of physical organization of data storage;
- indexing mechanisms (B-tree, hash).
- basics of cost-based optimization;
- transaction model and isolation levels.
- differences between relational and non-relational models.
- principles of storing and searching embeddings.
- architectural trade-offs of scalability and consistency.

Be able to:

- design the database schema.
- create and analyze indexes.
- interpret the request execution plan.
- perform performance measurements.
- reproduce and analyze transaction conflicts.
- choose a storage model for a specific task.
- design a hybrid storage architecture.
- critically evaluate LLM recommendations.

Be proficient in:

- experimental analysis methodology.
- culture of engineering measurement;
- skills of architectural justification;
- practical application of SQL, NoSQL,and vector storage.
- the method of using LLM as an engineering assistant and opponent.

6. Methodological concept of the discipline

6.1. Discipline as an engineering bridge

The course serves as a transition from algorithmic thinking to architectural thinking of storage systems.

It is based on the following principle:

Data Structure → Physical Implementation → Indexing → Optimization → Transactions → Architecture → Scaling → AI integration.

6.2. Master class as a form of lecture

Lecture classes are implemented in the format of an engineering master class.

Each master class includes:

- demonstration of measurement techniques.
- analysis of system behavior.
- reproduction of degradation with increasing load;
- comparison of alternatives.
- analysis of actual execution plans.

The lecture demonstrates a way of engineering thinking, not just a theoretical model.

6.3. Digital laboratory: Core of the Course

Laboratory classes are organized in a three-level structure:

1. Exercise - mastering the mechanism
2. Task-applying the mechanism
3. Engineering practical task-justification of the solution

Required elements of the laboratory program:

- measurement.
- logging;
- analysis.
- explanation of the result.

Failure to include measurements constitutes incomplete task completion.

7. The role and contribution of LLM in the educational process

LLM is integrated into the discipline as a structural element of the educational model.

7.1. LLM as a Tool for Experiment Design

Used for:

- formulating hypotheses.
- drawing up a measurement plan;
- generating test data.

Each hypothesis must be tested by experiment.

7.2. LLM as an analytical assistant

Used for analysis of:

- EXPLAIN-plans.
- transaction conflicts.
- index degradation.
- comparing architectures.

The final explanation is formulated by the student.

7.3. LLM as an architectural opponent

When designing an LLM storage architecture:

- identifies weaknesses.
- generates clarifying questions.
- offers alternative solutions.

This develops an engineering argument.

7.4. The principle of mandatory verification

Any output generated by LLM must be subject to:

- experimental verification;
- measurement.
- log analysis.
- interpretation of the execution plan.

The discipline is based on the following professional principles:
trust the measurement, not the wording.

8. Educational technologies

The discipline uses:

- master classes;
- digital laboratory tests;
- load testing;
- architectural case studies;
- LLM-based peer review.
- brief oral engineering interviews;
- defense of laboratory solutions.

9. Differentiated assessment model
and the principle of supportive ambition

Discipline is implemented with a high level of requirements.
This takes into account the heterogeneity of students' training.

9.1. Unified educational field

All students work with the same cases and tasks.
Differentiation is achieved through the depth of analysis and the degree of independence.

9.2. Development levels

Basic level

- correct implementation.
- availability of measurements.
- understanding the mechanism.

Advanced level

- comparative analysis of alternatives.

- architectural argumentation.

Research level

- additional experiments;
- scalability analysis.
- critical evaluation of LLM solutions.

9.3. Evaluation components

The assessment includes:

- engineering correctness.
- depth of analysis.
- quality of LLM-based reflection.
- oral defense.

10. Engineering certification

The exam is conducted in the format of defending an architectural case.
The student must:

- Design a data storage architecture for the AI system.
- justify the choice of the model.
- demonstrate an understanding of indexing and transactions.
- explain scalability considerations.
- answer the teacher's questions.

LLM can be used in preparation, but the final defense is conducted individually.

CALENDAR-THEMATIC SCHEDULE

Key:

Lecture (Lect.) — theoretical material delivery.

Practical / Lab. — hands-on activities, simulations and discussions.
Independent study (IS) — self-directed work on project tasks and analysis.
LLM use — applying Large Language Models for analysis, generation and decision support (integrated into independent study tasks).

Week.	Subject and content (including format and LLM)	Lect. (h)	Lab. (h)	IS / SDS (h)
1	**Persistent storage vs RAM. Physics of I/O** Master class: latencyRAM vs disk latency measurement, I/O profiling. Lab: (1) Measurements; (2) time growth graph; (3) implementation of the file index. LLM: experimental plan + hypotheses of degradation causes. SDS: report + LLM reflection.	2	2	2

Week.	Subject and content (including format and LLM)	Lect. (h)	Lab. (h)	IS / SDS (h)
2	**Page-based storage. Buffering** Master class: the impact of page size. Lab: page size emulation → I/O calculation → optimum selection. LLM: analysis of trade-offs, page size vs cost. SDS: a comparative report.	2	2	2
3	**Indexes: B+tree and hash** Master-class: EXPLAIN ANALYZE; full scan vs index scan. Lab: creating indexes; comparing plans; taking measurements. LLM: Parsing the execution plan. SDS: analytical report.	2	2	2
4	**Normalization and circuit design** Master class: demonstration of anomalies → 3NF. Lab: schema design; DDL; constraint testing. LLM: Schema audit. SDS: refactoring + LLM conclusion.	2	2	2
5	**SQL and JOIN algorithms** Master class: analyzing a suboptimal query. Lab: optimization; proof of acceleration. LLM: Comparison of alternative SQL queries. SDS: an analytical SQL case.	2	2	2
6	**Cost-based optimizer** Master class: statistics and plan selection. Lab: changing indexes; analysis of EXPLAIN. LLM: hypotheses of plan selection. SDS: a report with hypothesis testing.	2	2	2
7	**Transactions and isolation. Deadlock** Master class: dirty read and lock conflicts. Lab: deadlock playback; log analysis. LLM: step-by-step analysis of the conflict. SDS: case analysis.	2	2	2
8	**Document and key-value. CAP** Master class: SQL vs document on a single case. Lab: redesign of the circuit; performance measurements. LLM: Normalization vs denormalization. SDS: analytical note.	2	2	2
9	**Graph databases** Master class: SQL traversal vs graph query. Lab: graph construction; path finding; time comparison. LLM: justification for choosing a model. SDS: report.	1	2	2

Week.	Subject and content (including format and LLM)	Lect. (h)	Lab. (h)	IS / SDS (h)
10	**Embeddings and similarity search** Master-class: cosine similarity. Lab: search implementation; error analysis. LLM: false positivespositives/negatives analysis. SDS: interpretation of results.	1	2	2
11	**Vector DB and ANN** Master-class: brute force vs ANN. Lab: index setting; latency/quality comparison. LLM: Analysis of the speed/accuracy tradeoff. SDS: comparative report.	1	2	2
12	**ETL and data pipelines** Master-class: batch vs streaming. Lab: ETL implementation; logging. LLM: Data quality checklist. SDS: optimization of the pipeline.	1	1	2
13	**Replication and fault tolerance** Master class: failover. Lab: Crash behavior analysis. LLM: runbook recovery. SDS: report.	1	1	2
14	**Integration of SQL + NoSQL + Vector** Master class: Hybrid storage architecture. Lab: component integration; load testing. LLM: architectural audit. SDS: project preparation.	1	1	2
15	**Final engineering work** Master class: analyzing common errors. Lab: implementation of storage architecture; measurement; solution protection. LLM: expert verification based on the checklist. SDS: preparation for the exam.	1	1	3