Big Data (DS-GA 1004) – Lecture 2 Finals Preparation Notes

Fully based on Week 2 Slides + Lecture Transcript + Expanded Knowledge

Spring 2025

1 Centralized Systems: File Systems and Relational Databases

2 Announcements

- HW1 due next Monday (02/10).
- Lab 2 focuses on SQL.
- Updated syllabus available on Brightspace.

3 Confusion, Doubt, and Struggle (CDS) Questions

3.1 Is a large random array "data"?

Yes, if it is used for input to models, simulations, or analysis, even if it is synthetic.

3.2 What is a Transistor?

- Basic electronic switch: allows or blocks electrical current.
- Fundamental building block of CPUs.
- Miniaturization led to Moore's Law and modern computing.

3.3 What is a CPU?

- Central Processing Unit: Executes programs by processing data and instructions.
- Contains Arithmetic Logic Unit (ALU), Control Unit, and Registers.
- Modern CPUs have multiple cores (multi-core architecture).

3.4 How does a Computer Work?

- CPU interacts with memory (RAM) and storage (HDD/SSD).
- Data flows at speeds constrained by physical wiring and architecture.
- Minimizing distances (e.g., cache memory near CPU) improves speed.

4 File Systems

4.1 Overview

- Organizes data into hierarchical structures (directories and files).
- Persistent storage across sessions.

4.2 Advantages

- Simplicity.
- Portability.
- Low overhead.

4.3 Limitations

- Poor support for complex queries.
- Limited metadata.
- Difficult to enforce data consistency.

5 Relational Databases

5.1 Why Use Databases?

- Structured querying (SQL).
- Data integrity and schema enforcement.
- Concurrency management.
- Scalability for larger datasets.

5.2 Relational Model Concepts

- Data is stored in **tables** (relations).
- Each table consists of **tuples** (rows) and **attributes** (columns).
- Tuples are unordered and unique.

5.3 Mathematical Basis

- A relation R over sets A1, A2, ..., An is a subset of the Cartesian product $A1 \times A2 \times \cdots \times An$.
- $R \subseteq A1 \times A2 \times \cdots \times An$.

6 Structured Query Language (SQL)

6.1 Overview

- Declarative language: Describe *what* you want, not *how* to get it.
- Supported by almost all modern RDBMS.

6.2 Core SQL Operations

- `SELECT`: Retrieve data.
- `INSERT`: Add new data.
- `UPDATE`: Modify existing data.
- `DELETE`: Remove data.

6.3 Joins

- **INNER JOIN**: Matching rows only.
- **LEFT OUTER JOIN**: All from left table + matched from right.
- **RIGHT OUTER JOIN**: All from right table + matched from left.
- **FULL OUTER JOIN**: All rows from both, matched where possible.
- **CROSS JOIN**: Cartesian product.

6.4 Aggregation Functions

- `AVG`, `SUM`, `COUNT`, `MAX`, `MIN`.
- `GROUP BY` and `HAVING` clauses for grouping and filtering aggregates.

7 Indexing

7.1 Purpose

- Speeds up retrieval operations.
- Organizes data in structures like B-trees or hash maps.

7.2 Trade-offs

- Increases storage space.
- Slows down `INSERT` and `UPDATE` operations.

8 ACID Properties of Transactions

8.1 Atomicity

- All or nothing execution.
- Partial updates are rolled back.

8.2 Consistency

- Enforces database rules.
- Moves from valid state to valid state.

8.3 Isolation

- Transactions operate independently.
- Prevents "dirty reads," "non-repeatable reads," "phantoms."

8.4 Durability

- Once committed, the result persists even after crashes.

9 Expanded Knowledge: Real-World Concepts

9.1 CAP Theorem (Related to Distributed Databases)

- You can have at most two out of three: **Consistency**, **Availability**, **Partition Tolerance**.

9.2 Normalization vs. Denormalization

- **Normalization**: Remove redundancy, maintain integrity.
- **Denormalization**: Add redundancy to optimize read performance.

9.3 Common SQL Pitfalls

- **N+1 Query Problem**: Caused by poor join design.
- **Over-indexing**: Can degrade performance.
- **Non-optimal GROUP BY**: Can lead to memory overuse.

10 Summary

- File systems are simple but limited.
- Relational databases provide structure, consistency, and concurrency.
- SQL empowers data querying.
- Indexes accelerate reads but slow writes.
- ACID ensures transactional reliability.
- Normalization minimizes redundancy but may complicate queries.
- CAP theorem explains distributed system trade-offs.

11 Next Week

- Topic: **Distributed Computation with Map-Reduce**.
- Readings:
  - Dean & Ghemawat (MapReduce)
  - DeWitt & Stonebraker (Parallel Databases)