Seminar 6: About Lambda Architecture - https://www.oreilly.com/radar/questioning-the-lambda-architecture/
About Data Migrations - https://stripe.com/blog/online-migrations
Two articles discussing the pros and cons of lambda architecture, as well as an approach to run data migrations.
Keywords: CAP theorem, MapReduce, migration, *whatever word you didn't understand* etc.

- **What are the articles about?**

Those two papers describe a method for developing "big data" infrastructures that uses the lambda architecture to create applications around sophisticated asynchronous transformations that must execute with low latency and how one massive migration of hundreds of millions of objects may be done securely. It's also covered how to manage migrations and scale migrations.

- **What Lambda Architecture tries to solve?**

Organizations combine a standard batch pipeline with a quick real-time stream pipeline for data access in order to solve data processing problems. The Lambda Architecture tries to strike a balance between latency, data consistency, scalability, fault tolerance, and human fault tolerance, although it's been criticized for being unnecessarily complicated at times. Each pipeline has its own code base, which must be kept up to date to provide consistent, correct results when queries cross both pipelines. Re-computation and precomputation are also possible with this design. The Lambda Architecture's greatest benefit is that it eliminates human mistake tolerance and hardware damage tolerance.

- **What is the critique given against Lambda Architecture?**

An organization's use of lambda architecture to prepare for a data lake might have significant implications if no particular considerations are made:

1. Coding Big Data Frameworks is challenging.
2. The architecture's several levels may make it complicated, and synchronization between them can be costly, therefore it must be managed with caution.
3. Since of the separate and dispersed layers of batch and speed, support and maintenance of the architecture code can be challenging; moreover, maintenance of the architecture code might be difficult because it must give the same results in the distributed system.

- **Explain the keywords.**

1. **Theorem of the CAP**

Consistency, Availability, and Partition Tolerance (CAP) are acronyms for Consistency, Availability, and Partition Tolerance. According to the theory, a distributed system cannot always guarantee all three: consistency, availability, and partition tolerance. When things go wrong, it's necessary to make a trade-off and select at most two distributed system features to maintain.

2. **MapReduce**

MapReduce is a programming technique and software framework for processing large volumes of data.

3. **Migration**

The movement of data or software from one system to another is referred to as migration.

- **Explain the 4 steps of data migration.**

1. Dual writing: copy all data from the first table to the second table until it's completely copied, making sure to update both tables.
2. Changing all read paths: reading from the first table, comparing findings to those from the second table, and reading data from the second table.
3. Changing all write paths: writing all the data to both tables in reverse order while maintaining consistency in both tables.
4. Remove outdated data by ignoring tables that are no longer in use and shifting to a new table (the most recently used).

- **What can we achieve if we follow the pattern for data migration provided in the article?**

We may easily balance concerns about latency, data consistency, scalability, fault tolerance, and human fault tolerance on creating an application that employs "big data" by following this method.