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 articles describe an approach to designing "big data" infrastructures that uses the lambda architecture to build applications around complex asynchronous transformations that must run with low latency, as well as how one large migration of hundreds of millions of objects can be done safely. It's also covered how to handle migrations and how to do them at a large scale.

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

It is for the purpose of data access that businesses mix a typical batch pipeline with a quick real-time stream pipeline. The Lambda Architecture tries to strike a balance between latency, data consistency, scalability, fault tolerance, and human fault tolerance, although it has been criticized for being unnecessarily complicated at times. Each pipeline has its own code base, which must be kept in sync in order to deliver consistent, correct results when queries cross both pipelines. Re-computation and pre-computation are also possible with this design. One of the most significant advantages of the Lambda Architecture is that it eliminates human error tolerance as well as tolerance in the event of hardware failure.

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

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

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

- **Explain the keywords.**

1. **Theorem of CAP**

Consistency, Availability, and Partition Tolerance (CAP) are acronyms for Consistency, Availability, and Partition Tolerance, respectively. According to the theory, a distributed system can never guarantee all three of these characteristics: consistency, availability, and partition tolerance. When things go wrong, you must make a trade-off and maintain just two distributed system features.

2. **MapReduce**

MapReduce is a data-processing software framework and programming style.

3. **Migration**

The transfer of data or software from one system to another is referred to as a 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; make sure both tables are updated.
2. Changing all read paths: reading from the first table, comparing the findings with those from the second table, and reading data from the second table.
3. Changing all write paths: writing all data to both tables in reverse order, while maintaining consistency in both tables.
4. Delete outdated data by ignoring tables that are no longer in use and switching to a new one (the most recently used).

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

By using this pattern, we can easily balance latency, data consistency, scalability, fault tolerance, and human fault tolerance while developing a "big data" application.