Software Engineering Conference Russia 2018

October 12-13
Moscow

Business Intelligence in microservice architecture

Evgenii Vinogradov

Yandex.Money


SUN ODYSSEY 440

From Greece to Moscow

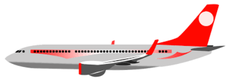
- Boat to Athens

- Feet to Taxi

11

- Athens Taxi to Airport

- Flight



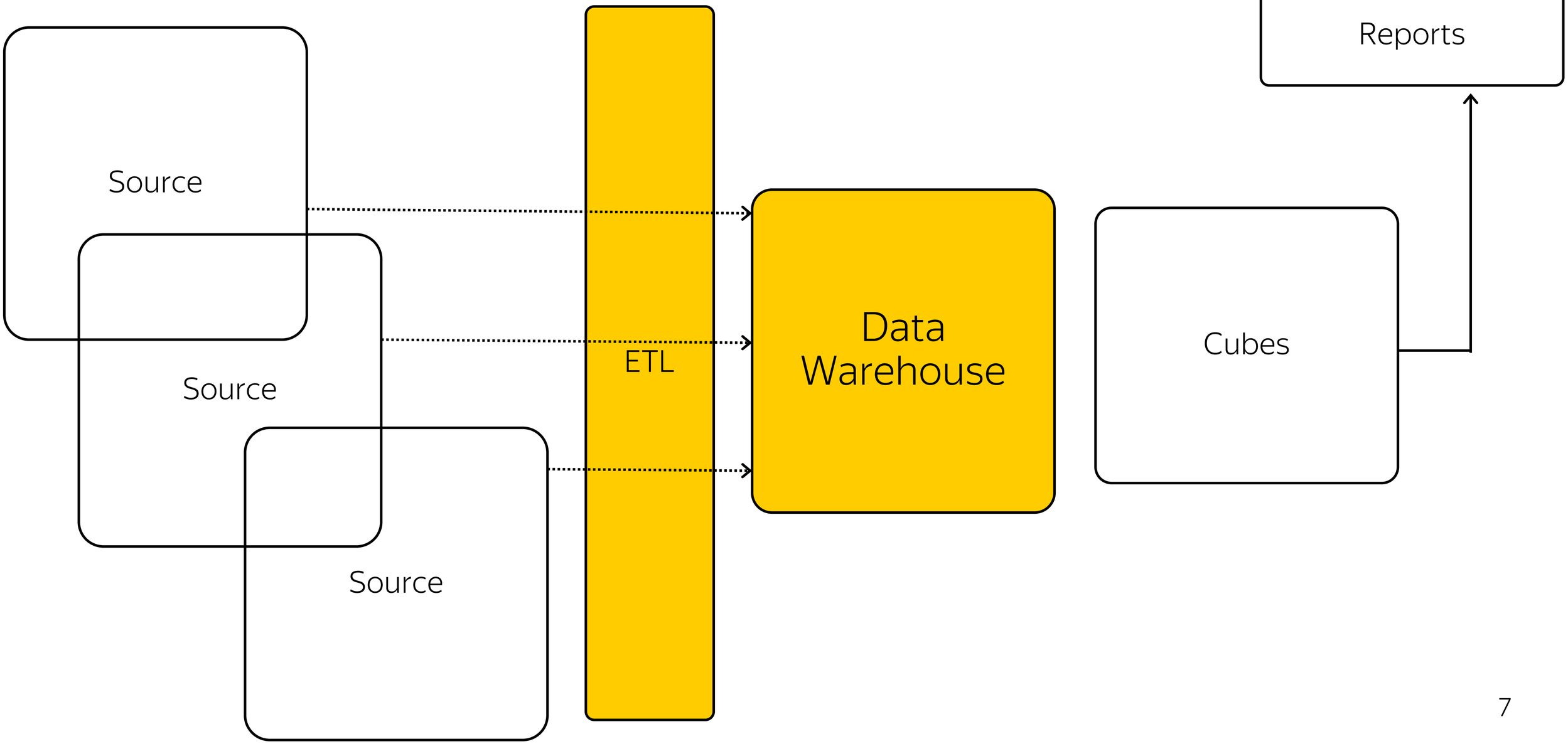
- Moscow Taxi to Digital October

- Profit!

How we built realtime BI

- High load
- Time to Show
- Request time
- Support architecture

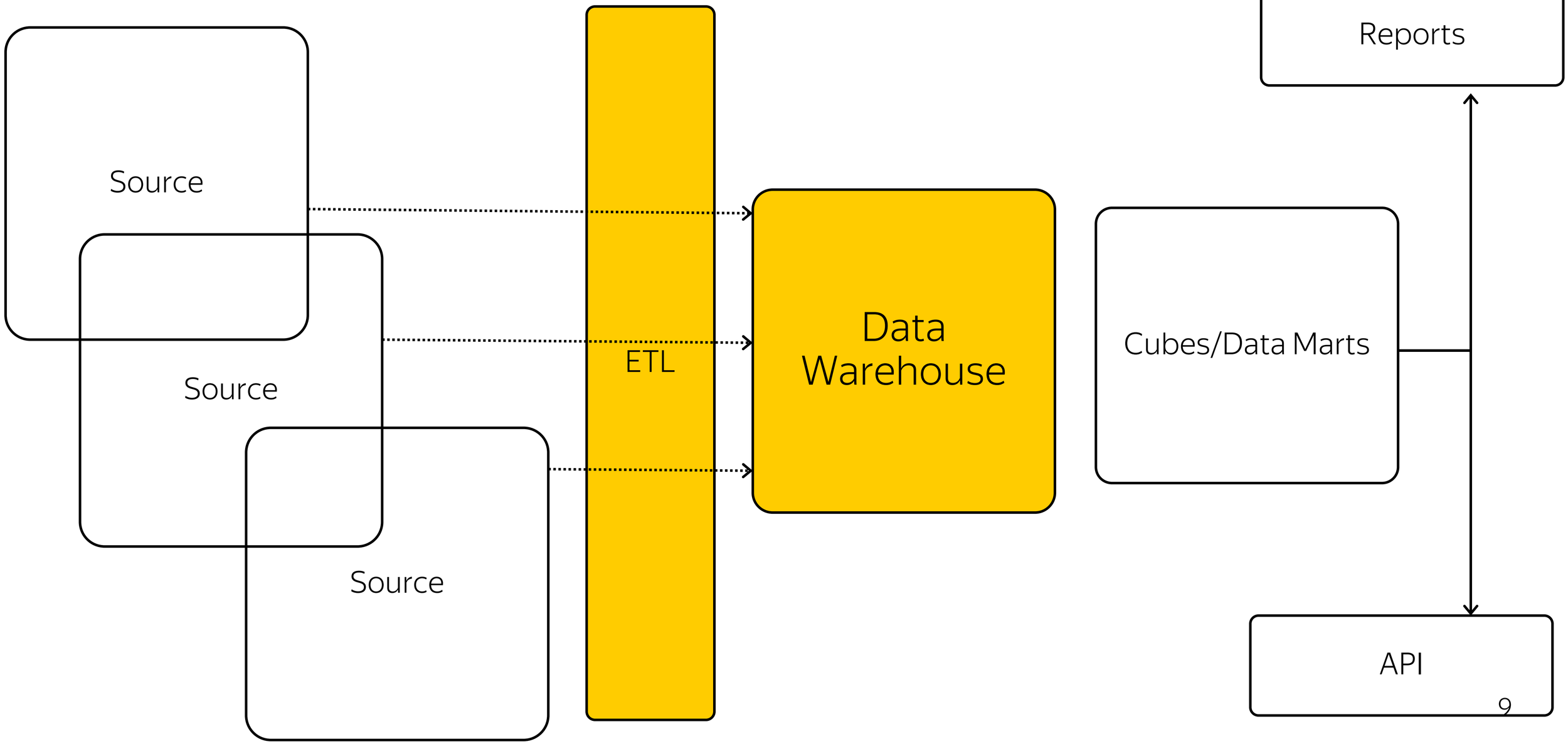
We start with ETL



We start with ETL

- Not so much sources
- Data is highly aggregated
- Gaps are rare and noticable

We start with ETL

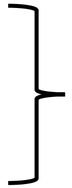


Then we add Backoffice interface

- More frequent ETL runs
- Source should be prepared

And SLA

- Data Update time
- Interface Update time
- Monitoring



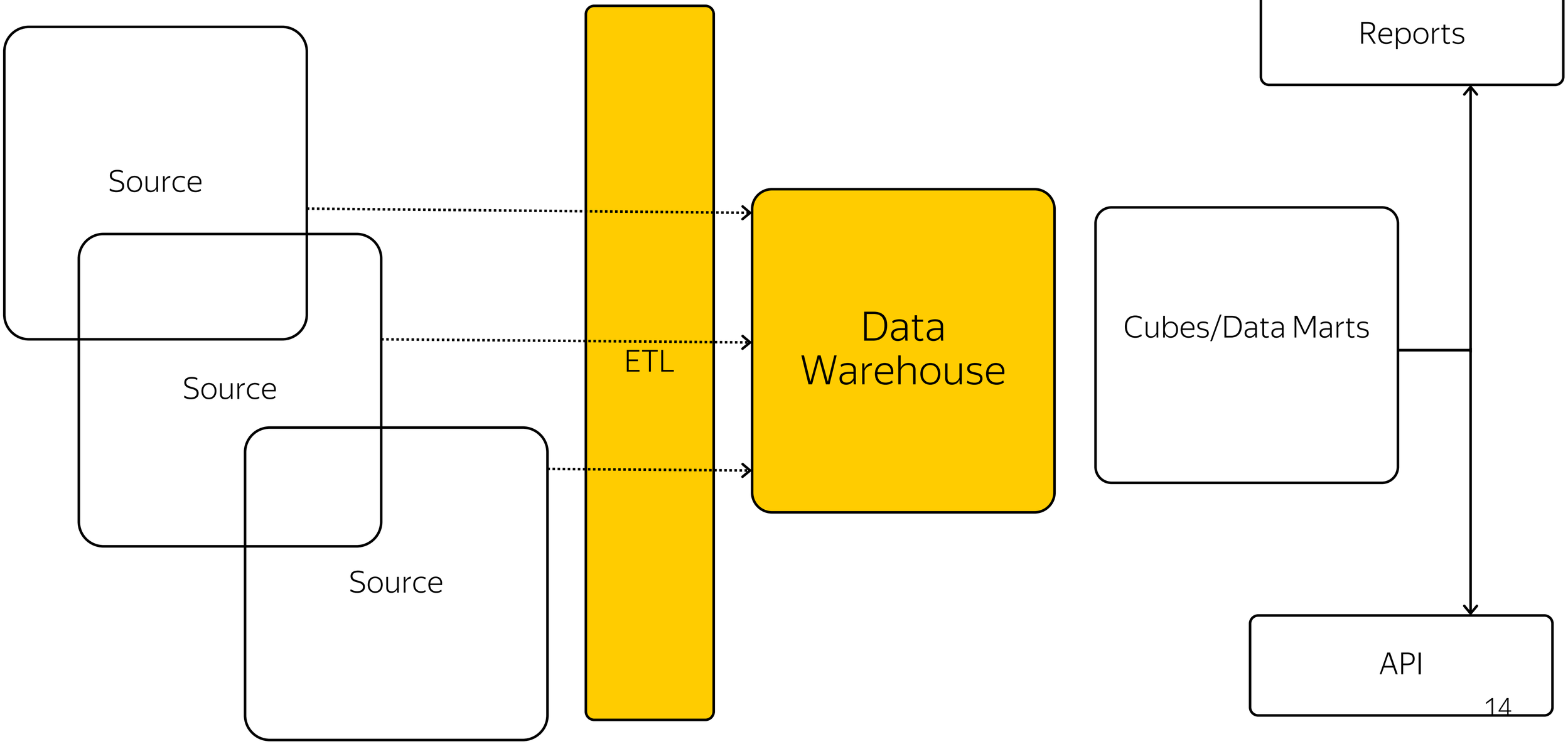
Time To Show

Testing

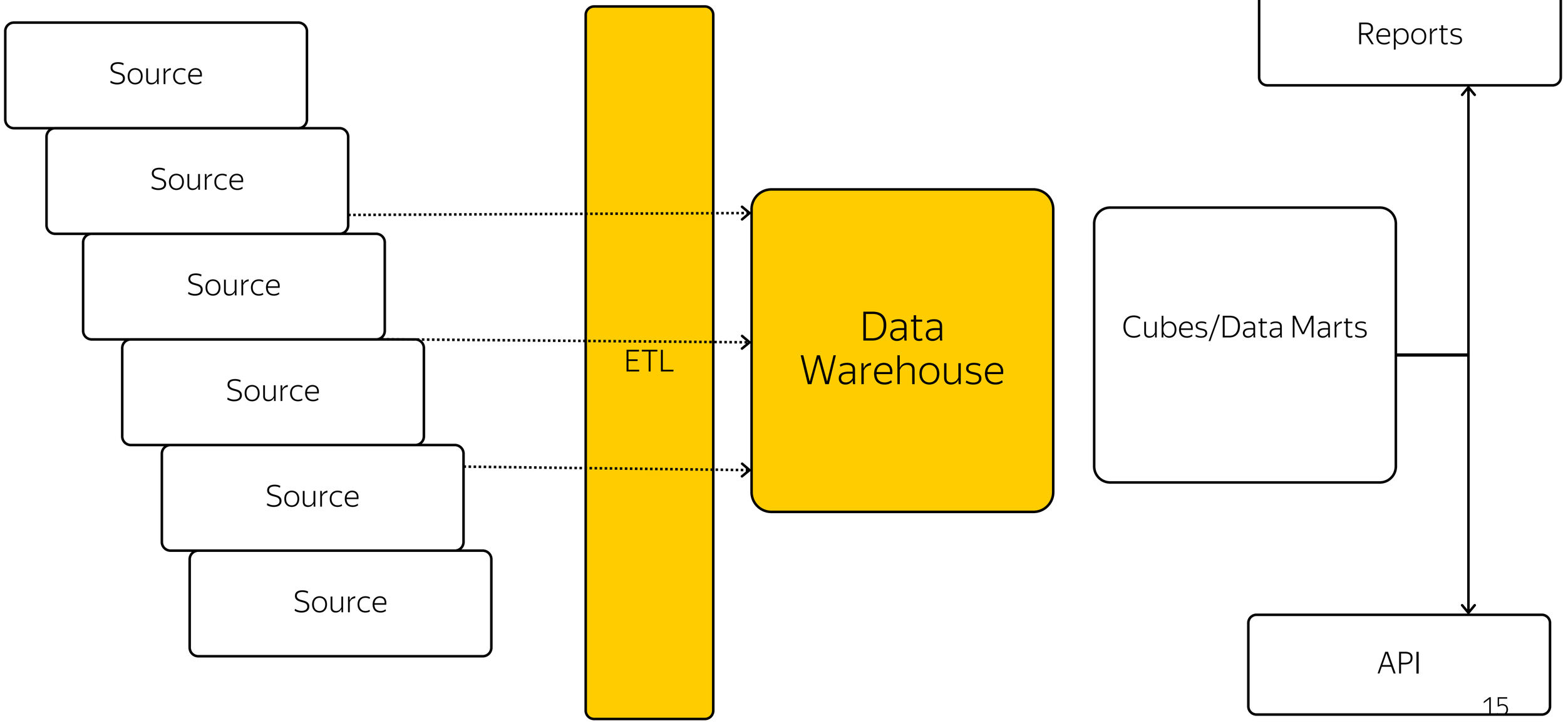
- Load testing – sources
- Load testing – ETL/API
- Load Testing – Together
- ETL – just pause
- And it is not so hard to create
- (You can test ETL with just pause)

Then we start microservicing

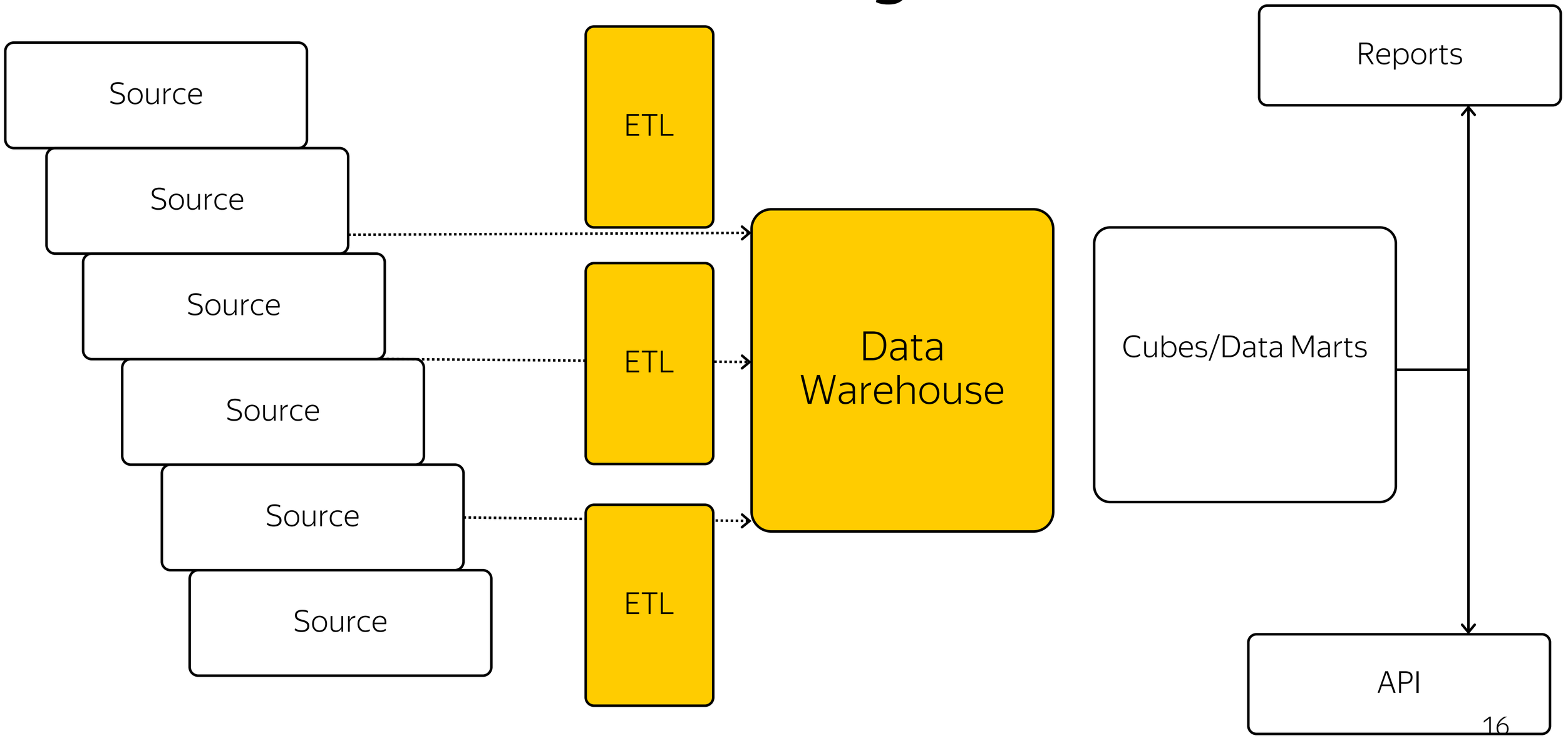
Then we start microservicing



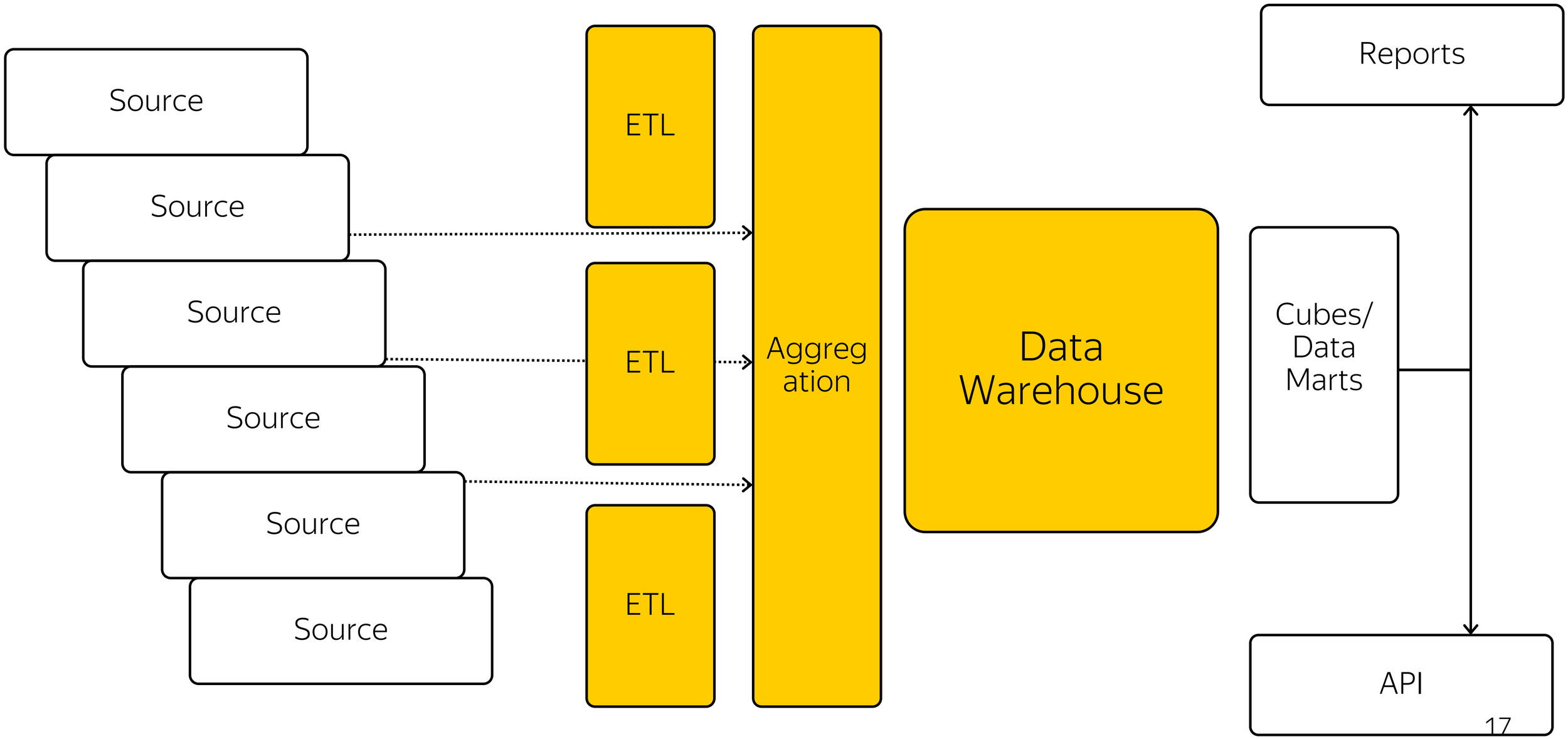
Then we start microservicing



Then we start microservicing



We start with ETL



Then we start microservicing

- A lot of sources give less aggregated and integrated data
- One (or more) aggregtation layer
- Data Update Time = [delay] * [sources per aggregation]

Then we start microservicing

- Unique Identifier Issue
- Aggregation rules sync
- ETL still is pretty good

Update strategies: trade request time for process time and load

- Load -> Store -> Show
- Load -> Aggregate -> Store -> Show
- Load -> Store -> Aggregate -> Show
- Load -> Aggregate -> Store - > Update -> Show
- Load -> Aggregate -> Store - > ~~Update~~ -> Show

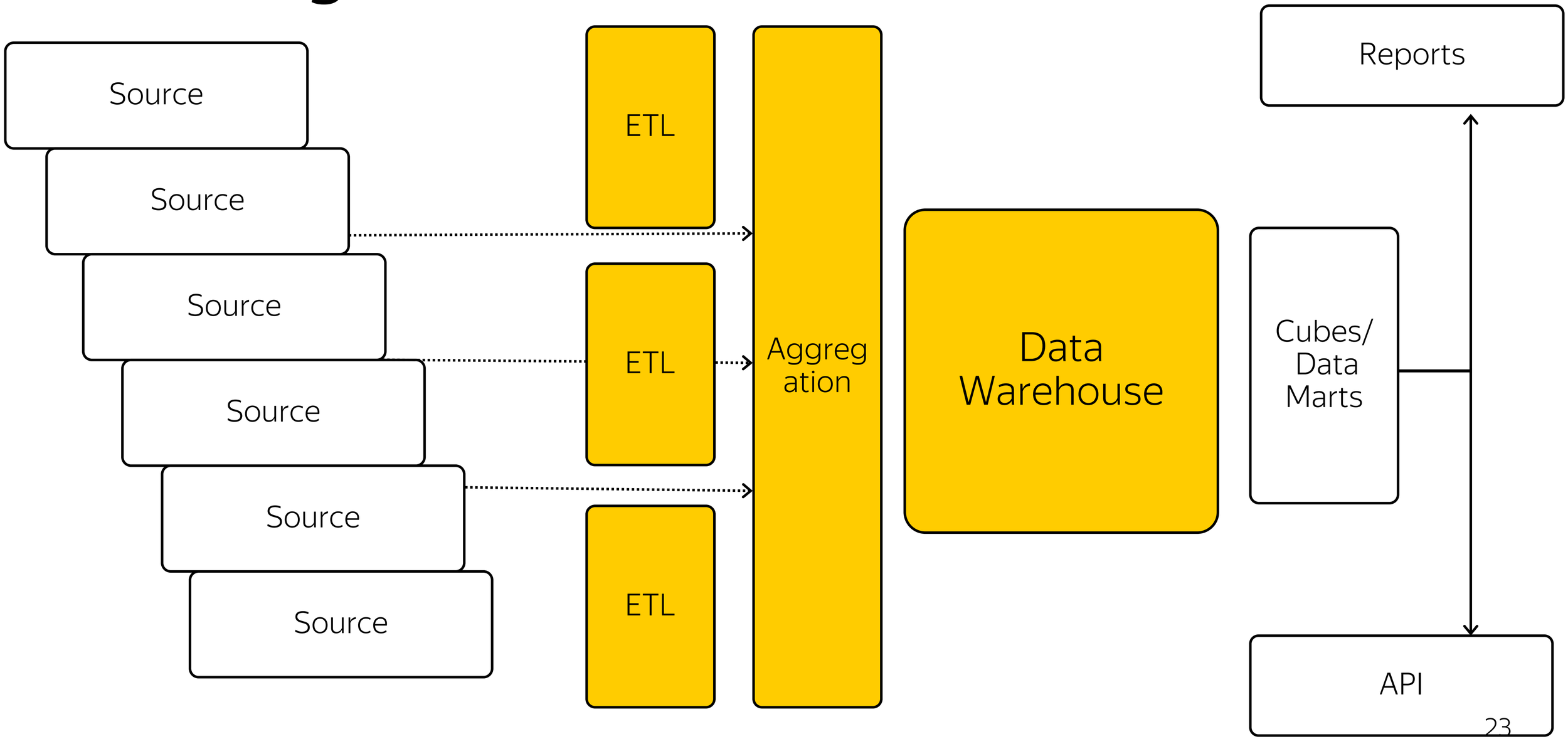
Update strategies: trade request time for process time and load

- Load -> Store -> Show
- **Load -> Aggregate -> Store -> Show**
- Load -> Store -> Aggregate -> Show
- Load -> Aggregate -> Store - > Update -> Show
- **Load -> Aggregate -> Store - > ~~Update~~ -> Show**

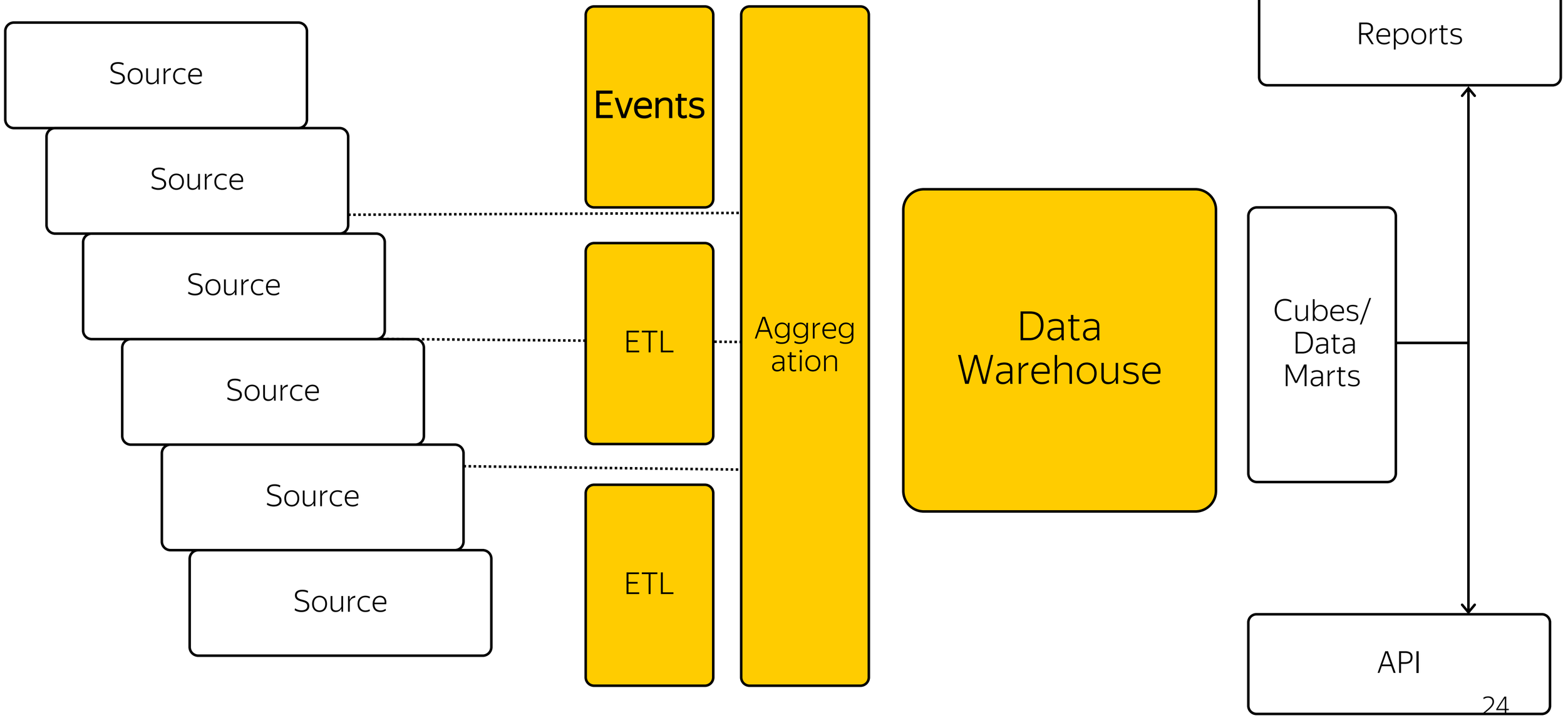
Drawbacks

- Sources still have to store a cache of data
- Sources have to preserve timelime (replication point)
- Request portion can be huge

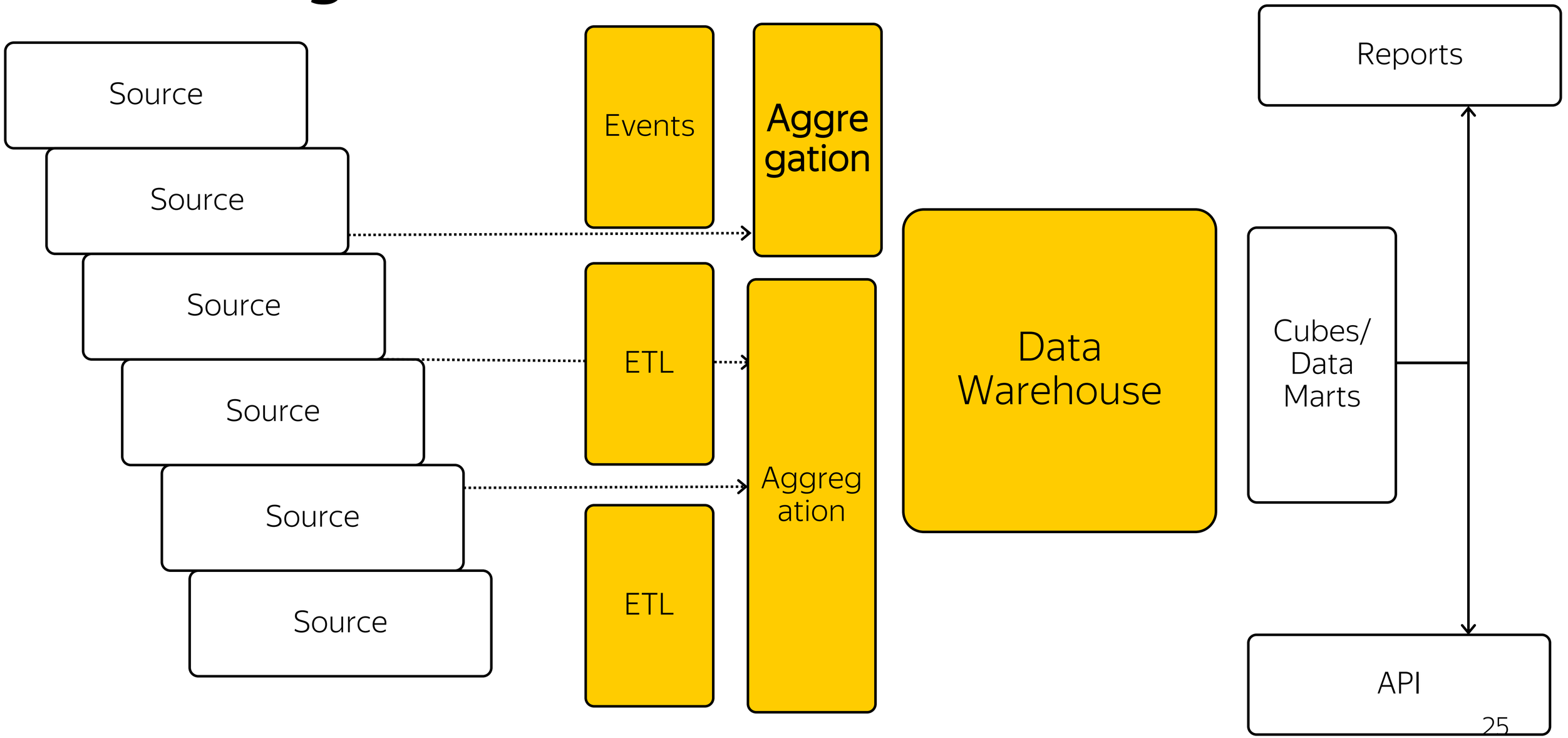
Then we go to events



Then we go to events



Then we go to events



Update strategies: trade request time for process time and load

- Load -> Store -> Show
- Load -> Aggregate -> Store -> Show
- Load -> Store -> Aggregate -> Show
- Load -> Aggregate -> Store - > Update -> Show
- Load -> Aggregate -> Store - > ~~Update~~ -> Show
- Load -> Construct -> Aggregate -> Store -> Show

11

Now

- We still have ETL for aggregations
- Time To Show: 10 sec / source
- Source type count: close to 100

Starring

- Data Warehouse: MS SQL Server
- ETL: SSIS
- Events queue: Kafka
- Events reader: C#
- Other Sources: API/Java/etc
- Out of scope: sharding, downtime, processing, etc

Key takeaways

- ETL works fine with Events
- Find sufficient unique identifier among data sources
- Metrics: Time to Show and others

Questions?

Evgenii VInogradov

Head of BI development

 jonny@yamoney.ru

 dm_jonny