Schema-on-read

(Credit: Krzysztof Rozacki)

Traditionally, schema-on-write has been used to load data into relational databases and data warehouses. Schema had to be defined before loading data into the database store. The reason for that was because storage and processing were treated as one and how data would be queried had to be defined before loading data. In the new world, storage and processing are separated. It's because:

1. In the Big Data world it is less expensive to move the processing to data, than data to the processing
2. There are many ways to query the data. The way one wants to query the data can be defined after data is in the storage and can be changed at will. Depending on how we treat data, we use different engines. Data can be treated as:
   a. tables: SQL
   b. text: Solr, Elastic
   c. graph: JanusGraph, Neo4j
   d. time series: HBASE
   e. documents: MongoDB
   f. keys and values: HBASE
   g. matrixes: TileDB
3. In many cases, it would be very constraining, if not impossible, to try to define the precise schema for current and future data.

Another useful feature of splitting storage from processing is the ingestion performance. If the schema is defined upfront, the speed at which data lands in the database is slower when compared to the speed at which data is ingested without the schema. It's important when:

- there is a lot of data to consume or we dealing with streams of data
- we have limited time to ingest

Schema-on-read concept is used across the entire Big Data stack:

- Kafka: PubSub messaging storage
- HBASE: database, a distributed, scalable, big data store
- Solr: Text search database