DHBW Ravensburg

Intelligence Engineering

Solving Real World Problems At Scale

Prof. Dr.-Ing. Mark Schutera | November 5, 2026 · *drifting stream*

LECTURE III

Data Engineering Fundamentals

This lecture in one slide

III Data Engineering Fundamentals

Data Sources
Data Formats
Data Models
Data Storage Engines and Processing
Modes of Dataflow

DATA
ENGINEERING
FUNDAMENTALS

... or knowing how to collect, process, store, retrieve and process an increasing amount of data.

Knowing how to handle a growing amount of data

Data Sources
DATA ENGINEERING FUNDAMENTALS
... or knowing how to collect, process, store, retrieve and process an increasing amount of data.

Data Sources
Data Formats, Structures, & Models
DATA ENGINEERING FUNDAMENTALS
... or knowing how to collect, process, store, retrieve and process an increasing amount of data.

Data Sources
Data Formats, Structures, & Models
DATA ENGINEERING FUNDAMENTALS
Data Storage, Retrieval & Processing.
"... or knowing how to collect, process, store, retrieve and process an increasing amount of data."

Further reading: Kleppmann 2017

1

Data Sources

DATA SOURCES
Characteristics
Purposes
Processing Methods

Knowing your data lets you use it efficiently

Common practice to log everything you can.
... and you can store.
... and are allowed to.

Log everything you can, store, are allowed to

Common practice to log everything you can.

... and you can store.

... and are allowed to.

Uses Input Data

Malformatted, any format, requires fast processing.

Common practice to log everything you can.
... and you can store.
... and are allowed to.
Uses Input Data
Halfformatted, any format, requires fast processing.
System Generated Data
Logs, System outputs, model predictions.

Common practice to log everything you can.

... and you can store.

... and are allowed to.

User Input Data

Halfformatted, any format, requires fast processing.

System Generated Data

Logs, System outputs, model predictions.

Internal Databases

Enterprise applications and assets.

Common practice to log everything you can.
... and you can store.
... and are allowed to.

User Input Data
Halfformatted, any format, requires fast processing.

System Generated Data
Logs, System outputs, model predictions.

Internal Databases
Enterprise applications and assets.

Third Party Data
Data collected on the public. As opposed to customers [1][2]. Apps, Websites, "Public Spaces", „the Internet".

Third party data is usually sold

Third Party Data

"Data collected on the public. As opposed to customers (1st). Apps, websites, "public spaces", "the internet".

€

Usually sold after being cleaned and processed.

"Data privacy vs. Fingerprinting.

#Apple vs. China.

2

Data Formats

DATA FORMATS

to pick a format, arouses the question on how the data will be used in future.

To pick a format, ask how the data will be used

DATA FORMATS.

to pick a format, answer the question on how the data will be used in future.

Then match the format based on the access pattern, meaning how a system will read or write your data.

DATA FORMATS.

to pick a format, answer the question on how the data will be used in future.

Then match the format based on the **access pattern**. Meaning how a system will read or write your data.

JSON
CSV
PARQUET
Avro
Protobuf
Pickle
!

DATA FORMATS.

to pick a format, answer the question on how the data will be used in future.

Then match the format based on the access pattern. Meaning how a system will read or write your data.

"Key": "Value" Paradigm
is trivial and versatile.
JSON
JAVA SCRIPT OBJECT NOTATION

"Key" : "Value" Paradigm
is trivial and versatile.
(ones)
adaptable
structuredness
and types.
JSON
JAVA SCRIPT OBJECT
NOTATION
{
"name" : "Mark",
"isVoting": true,
"what": {
....

Language independent, even human readable

"Key": "Value" Paradigm is trivial and versatile.
(ones) adaptable structuredness and types.
JAVA SCRIPT OBJECT NOTATION.
{
"name": "Mark",
"isWorking": true,
"what": {
....
(anyone) independant even human readable.
.... so they also take up quite some space.
#text vs. binary

CSV: consecutive elements side by side in memory

Row major: fast access to samples

#numpy ndarray default.
CSV COMMA-SEPARATED VALUES.
Consecutive Elements are stored next to each other in memory.
fast access.
MAJOR
Feature 1
Feature 2
Feature 3
Sample 1
Sample 2
usually better in sample writing scenarios.

#numpy ndarray default.
CSV COMMA-SEPARATED VALUES.
Consecutive Elements are stored next to each other in memory.
fast access.
Feature 1
Feature 2
Feature 3
Sample 1
Sample 2
MAJOR
usually better in sample writing scenarios.
fast access to features.
MAJOR
PARQUET
Consecutive Elements in a column are stored next to each other.

#numpy ndarray default.
CSV COMMA-SEPARATED VALUES.
Consecutive Elements are stored next to each other in memory.
fast access.
ROW-MAJOR
Feature 1
Feature 2
Feature 3
Sample 1
Sample 2
usually better in sample writing scenarios.
fast access to features.
ROW-MAJOR
PARQUET
Consecutive Elements in a column are stored next to each other.
#R & Pandas Dataframes.

Text files: assume each character is one byte

Binary files need knowledge to interpret

Binary: 2x faster to load, 6x more compact

3

Data Models

Data models: how data is represented

Data models affect what your system can solve

RELATIONAL.

each relation is a set of tuples, which are not ordered.

„Think table."

#CSV, Parquet.

NoSQL or nonrelational
or not only relational.
RELATIONAL.
each relation is a
set of tuples, which
are not ordered.
Document Model
. Same format: ID. "
Think, just rows.
"Everything in one place.
„Think table."
CSV, Parquet.
Graph Model
Relationships modeled
explicitly.

QUICK SIDEQUEST ON QUERY LANGUAGE: SQL.

A declarative language, or „you specify the output you want", not the steps that would take you there (imperative).

DATA MODELS are a pain to maintain, extend or migrate. Unstructured data doesn't adhere to a predefined schema.

Unstructured data has no predefined schema

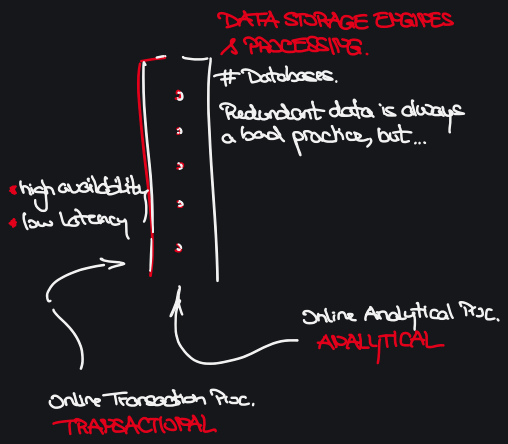
DATA MODELS are a pain to maintain, extend or migrate. unstructured data doesn't adhere to a predefined schema.
But it usually still contains valuable patterns and can be stored effectively #Idue.

4

Data Storage Engines and Processing

Databases: redundant data is a bad practice, but


DATA STORAGE ENGINES & PROCESSING.
#Databases.
Redundant data is always a bad practice, but...
high availability
low latency
Online Transaction Proc.
TRANSACTIONAL


DATA STORAGE ENGINES & PROCESSING.
Databases.
Redundant data is always a bad practice, but...
high availability
low latency
Online Analytical Proc.
ANALYTICAL
Online Transaction Proc.
TRANSACTIONAL

Online Transaction Proc.

TRANSACTIONAL

Atomicity.

Transaction fails with a single failed step.

Online Transaction Proc.

TRANSACTIONAL

- Atomicity.
  Transaction fails with a single failed step.
- Consistency.
  All transactions follow the same rules

Online Transaction Proc.

TRANSACTIONAL

Atomicity.
Transaction fails with a single failed step.

Consistency.
All transactions follow the same rules

Isolation.
Two transactions don't interfere with each other.

Online Transaction Proc.

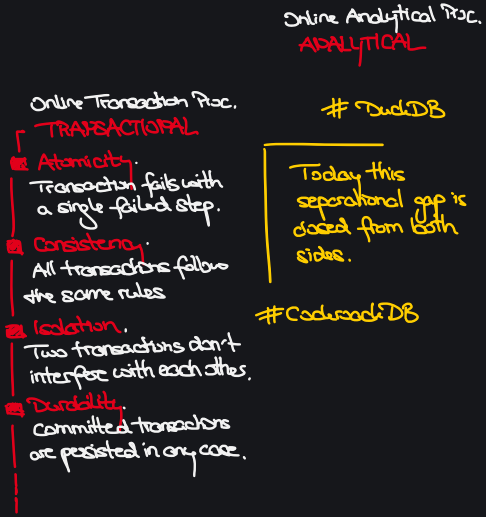
TRANSACTIONAL

Atomicity.
Transaction fails with a single failed step.

Consistency.
All transactions follow the same rules

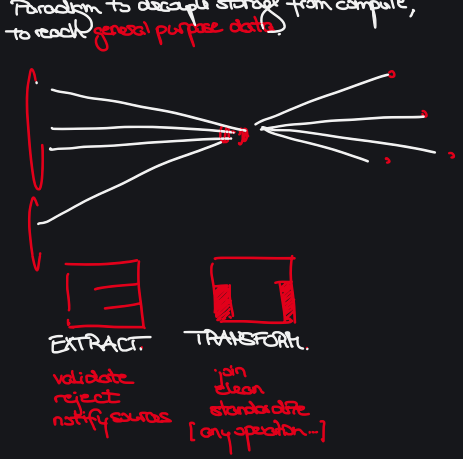
Isolation.
Two transactions don't interfere with each other.

Durability.
Committed transactions are persisted in any case.

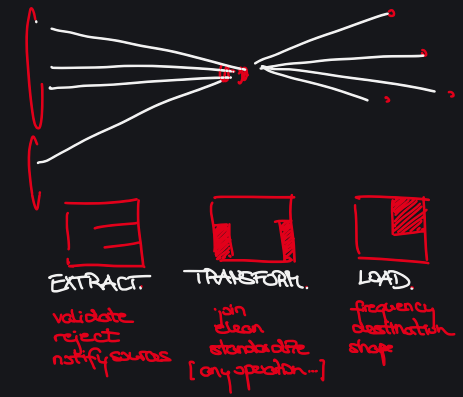

Online Transaction Proc.
TRANSACTIONAL
Atomicity.
Transaction fails with a single failed step.
Consistency.
All transactions follow the same rules
Isolation.
Two transactions don't interfere with each other.
Durability.
Committed transactions are persisted in any case.
Online Analytical Proc.
ANALYTICAL
#DuckDB
Today this separational gap is closed from both sides.
#CockroachDB

Paradigm to decouple storage from compute,
to reach general purpose data.

Paradigm to decouple storage from compute, to reach general purpose data.


Paradigm to decouple storage from compute,
to reach general purpose data.
EXTRACT.
validate
reject
notify sources
TRANSFORM.
join
clean
standardize
[any operation...]

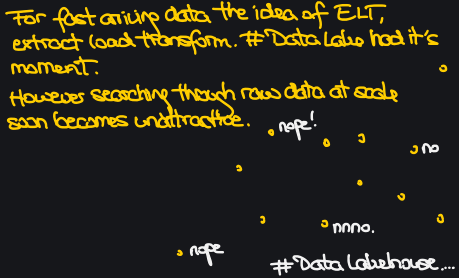
Paradigm to decouple storage from compute, to reach general purpose data.



The structure of most ML data layers

The underlying structure of the data layer of many (most) ML companies. While it requires to keep data sources structured, this allows to scale efficiently.

For fast arriving data the idea of ELT, extract load transform. #Data Lake had it's moment.
However searching through raw data at scale soon becomes unattractive.

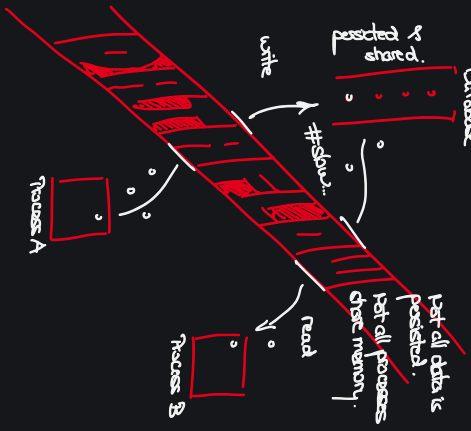

For fast arriving data the idea of ELT, extract load transform. #Data Lake had it's moment.
However searching through raw data at scale soon becomes unattractive.
nope!
no
nnno.
nope
#Data Lakehouse...

5

Modes of Dataflow

Not all data is persisted, not all memory shared

Write to a database: persisted and shared



Read from the database, slow

SERVICE-ORIENTED DESIGN.
Not all data is persisted.
Not all processes share memory.
Process B
Process A

SERVICE-ORIENTED DESIGN.
Not all data is persisted.
Not all processes share memory.
Just a shared network.
request.
expose.
Process A
Process B

More processes, more requests

Microservice Architecture
as a design principle.
Develop, test, maintain
independently. #API-first. #REST

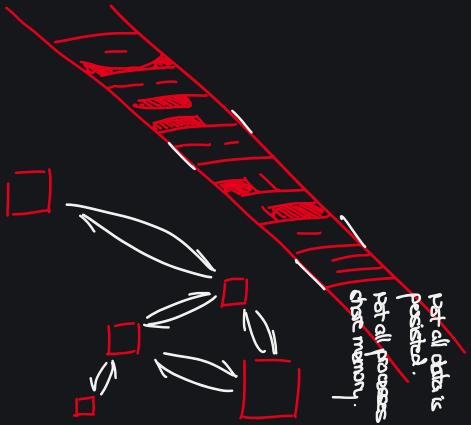
Microservice Architecture
as a design principle.
Develop, test, maintain
independently. #API first. #REST

FURTHER READING:
Building Microservices by Sam Newman.

Microservice Architecture
as a design principle.
Develop, test, maintain
independently. #API first. #REST

FURTHER READING:
Building Microservices by Sam Newman.

... and then design your org chart around your architecture.



Many processes, many data paths

Passing data gets complicated and stale fast

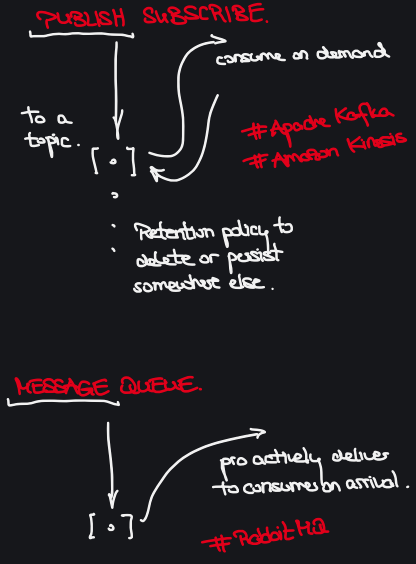
EVENT-DRIVEN.
Not all data is persisted.
Not all processes share memory.
DATAFLOW

Processes publish events to a bus

The broker, usually in memory

Publish subscribe: to a topic, consume on demand

Retention policy. Apache Kafka, Amazon Kinesis


PUBLISH SUBSCRIBE.
consume on demand
to a topic.
#Apache Kafka
#Amazon Kinesis
Retention policy to delete or persist somewhere else.
MESSAGE QUEUE.
pro actively deliver to consumers on arrival.
#RabbitMQ

This is all for streaming data, as opposed to historical, batch processed data.

References

1 Chip Huyen. *Designing Machine Learning Systems: An Iterative Process for Production-Ready Applications*. Sebastopol, CA: O'Reilly Media, 2022.

2 Martin Kleppmann. *Designing Data-Intensive Applications*. Sebastopol, CA: O'Reilly Media, 2017.

3 Sam Newman. *Building Microservices: Designing Fine-Grained Systems*. 2nd ed. Sebastopol, CA: O'Reilly Media, 2021.