



Artificial Intelligence and Data Analytics for Engineers (AIDAE)

Lecture 1
May, 22nd

Andrés Posada

Anas Abdelrazeq

Today's Lecturer

Marco Kemmerling

Vladimir Samsonov

Artificial Intelligence and Data Analytics for Engineers

Overview Lectures 1 – 4



1

Introduction to Data Analytics and Artificial Intelligence in Engineering: Organizational matters (e.g. exam, exercises, dates). Goals, Challenges, Obstacles, and Processes.



2

Introduction into the primary programming language of the lecture, Python: Syntax, libraries, IDEs etc. Why is Python the *lingua franca* of the Data Scientist?



3

Data Preparation: Cleansing and Transformation. How do real world data sets look like and why is cleaning and transformation an integral part of a Data Scientist's workflow?

4

Data Integration: Architectures, Challenges, and Approaches. How can you integrate various data sources into an overarching consolidating schema and why is this important?

Data Integration

Challenges

Fundamentals

Databases

Learning Objectives



Learning Objective
w.r.t. Knowledge/Understanding.

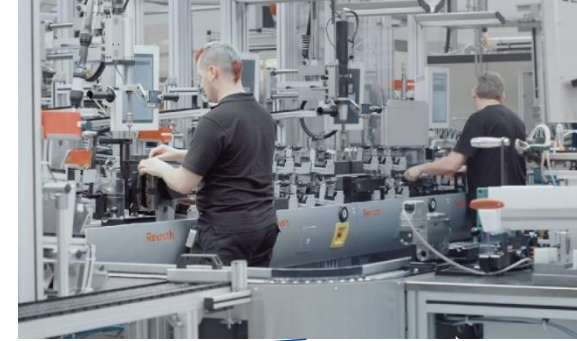
After successfully completing this lecture, the students will have achieved the following learning outcomes:

- Have an understanding of the scope and challenges of data access/data integration.
- Know about the fundamentals of databases.
- Know about different concepts, tools and methods for successful data access/data integration.

Motivation and Introduction

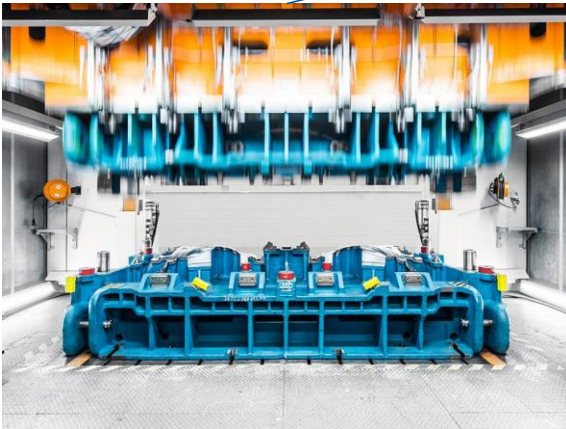
Starting Point

https://www.geartechnology.com/articles/0916/Managing_Shop_Floor_Data



?

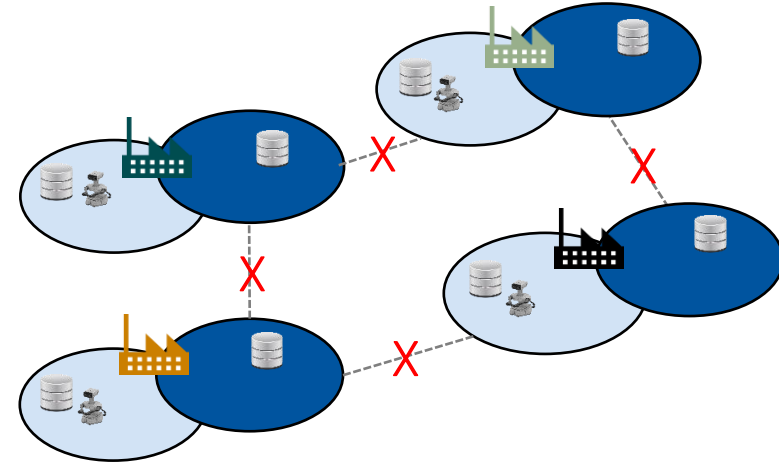
Any given data analysis task might require a variety of data sources to accomplish it!



An Industry Example from our Institute (Big, Company-Wide)

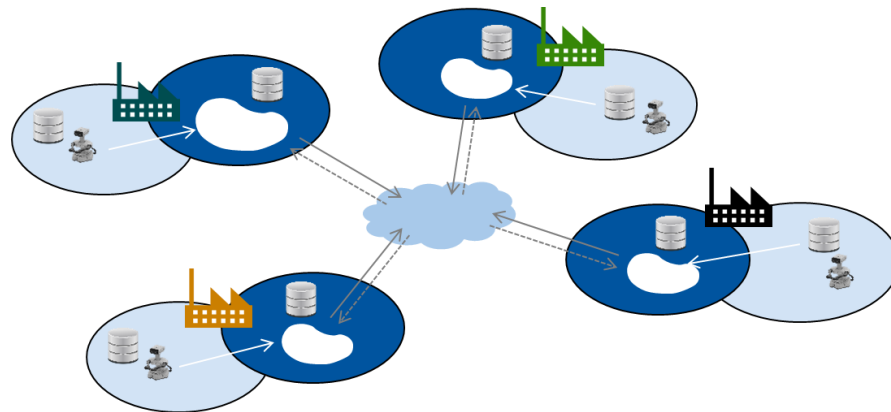
Initial Situation

- Many different data systems in different nets and positions
- Isolated stand-alone solutions enable only specific local data access
- No transparency over locations
- Optimization potential of an integrated perspective



Final Situation

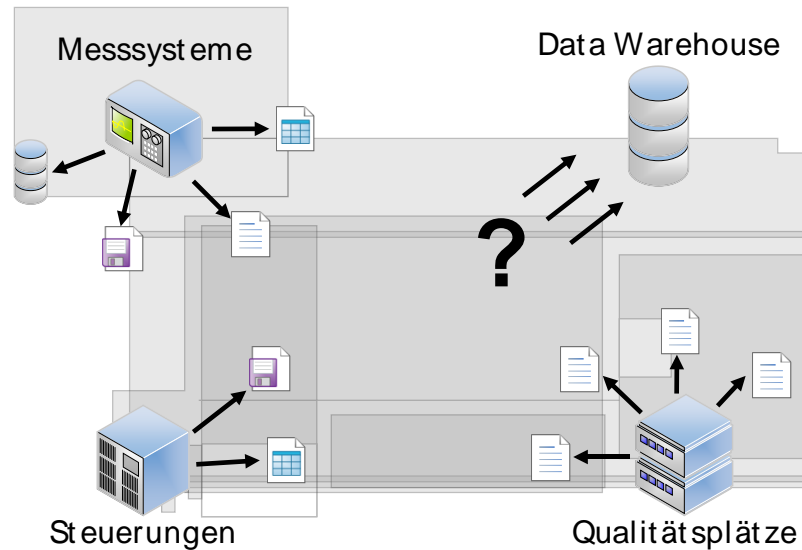
- Cross-plant transparent consolidation
- One homogenous view



An Industry Example from our Institute (Small, Department-Wide)

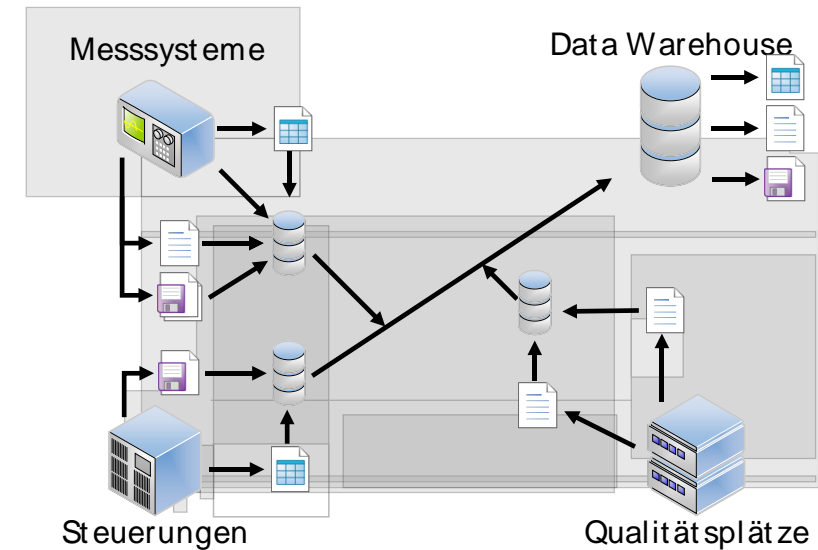
Initial Situation

- Many different data systems
- Heterogeneous technologies
- No homogeneous access point

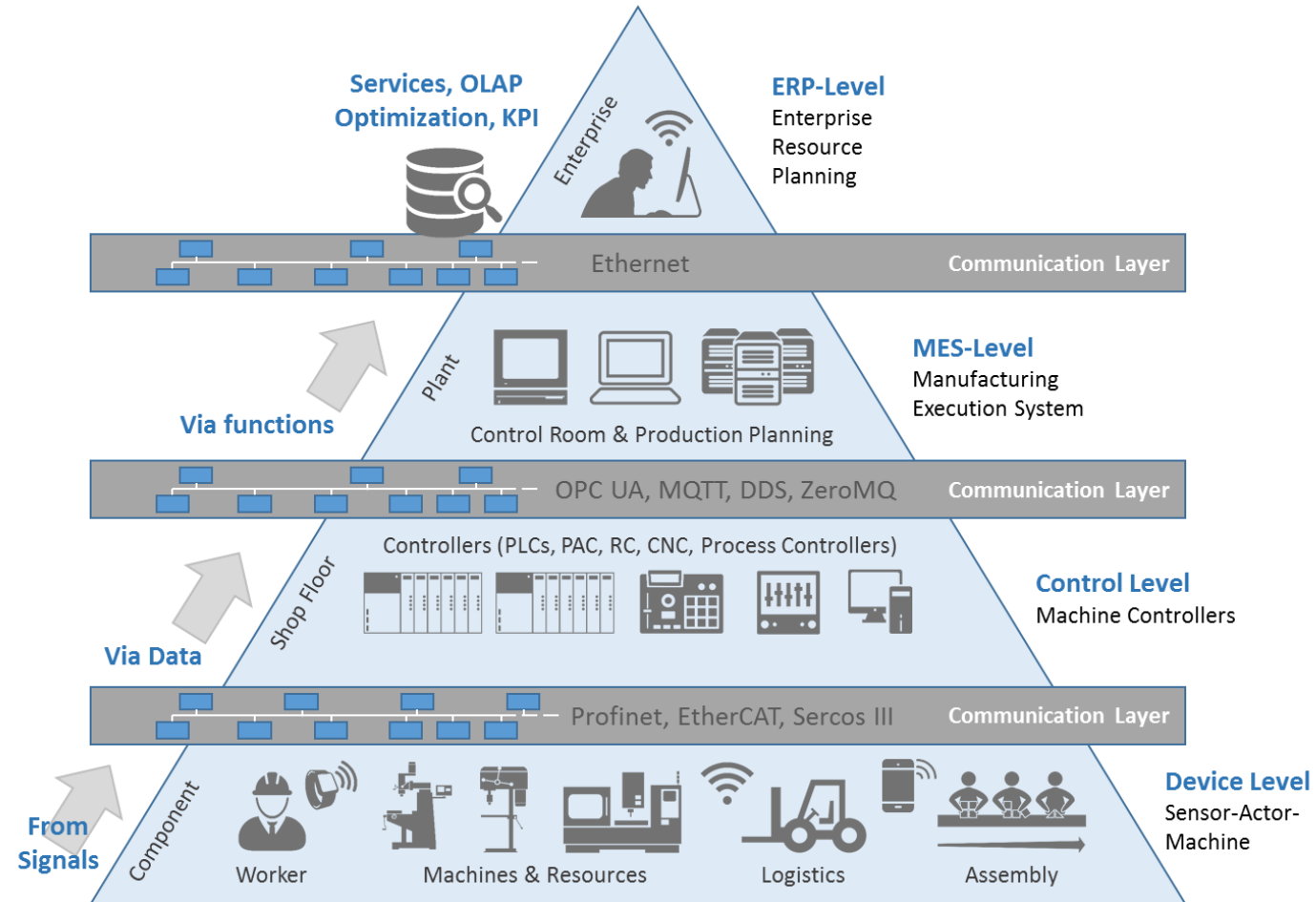


Final Situation

- One consolidated system architecture
- One homogenous view



Modern Engineering Environments (The Automation Pyramid)

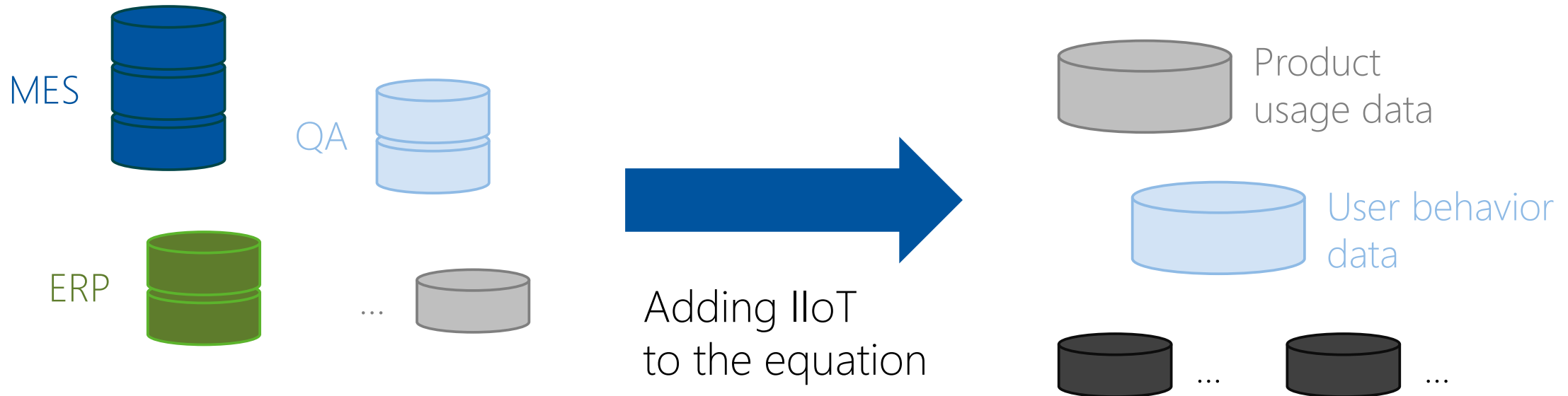



Munz, H. (2015): Requirements for Time Sensitive Networking. Why right now? Because Industry 4.0 needs it! IEEE 802.1 TSN Standard Meeting. KUKA. IEEE. 22.05.2015. URL: <http://www.ieee802.org/1/files/public/docs2015/tsn-munz-requirements-for-tsn-in-manufacturing-0515-v01.pdf>.

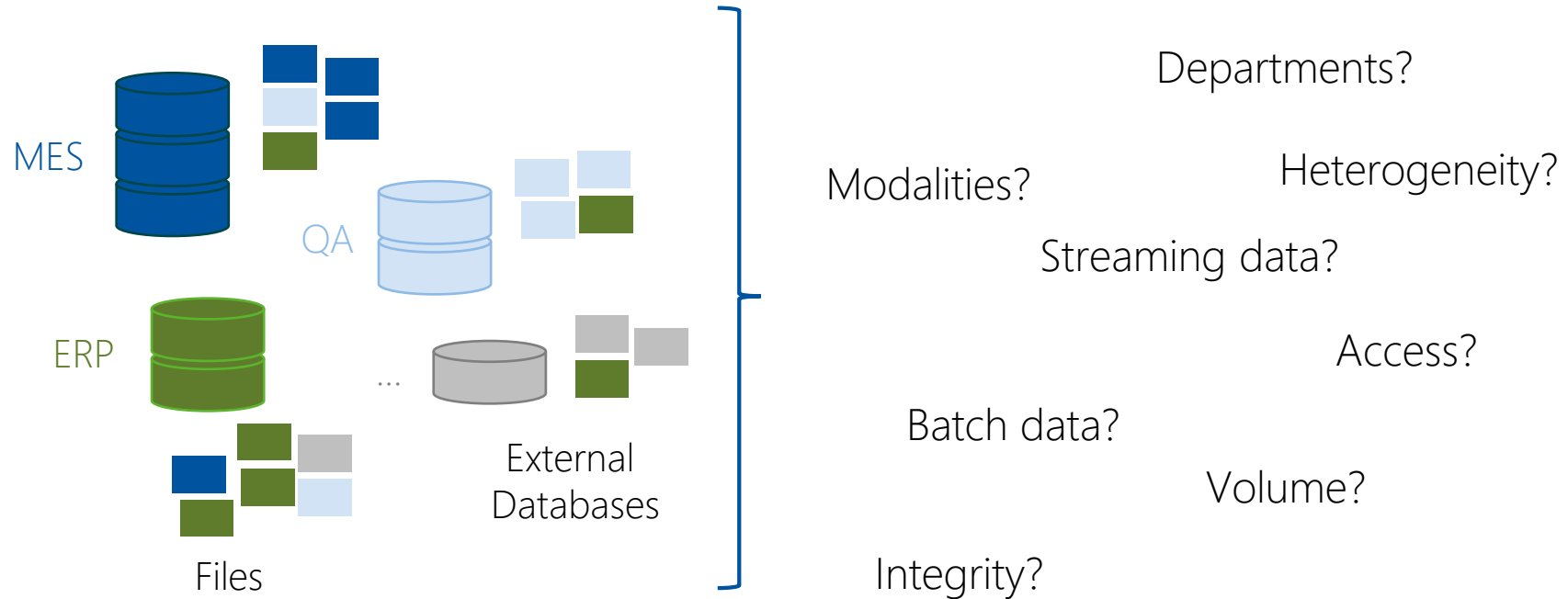
The Last Two Decades



Taking a look at the last 20 years of networking and data processing in industry, they focused on automation and optimization of specific business core processes...
... as a result several data silos and dispersed data stores have been grown throughout decades.



- 
- Lessons learned:** you do not start with data analytics itself!
- Most of the time the first challenge is to **realize the data availability** and the IT-infrastructures.



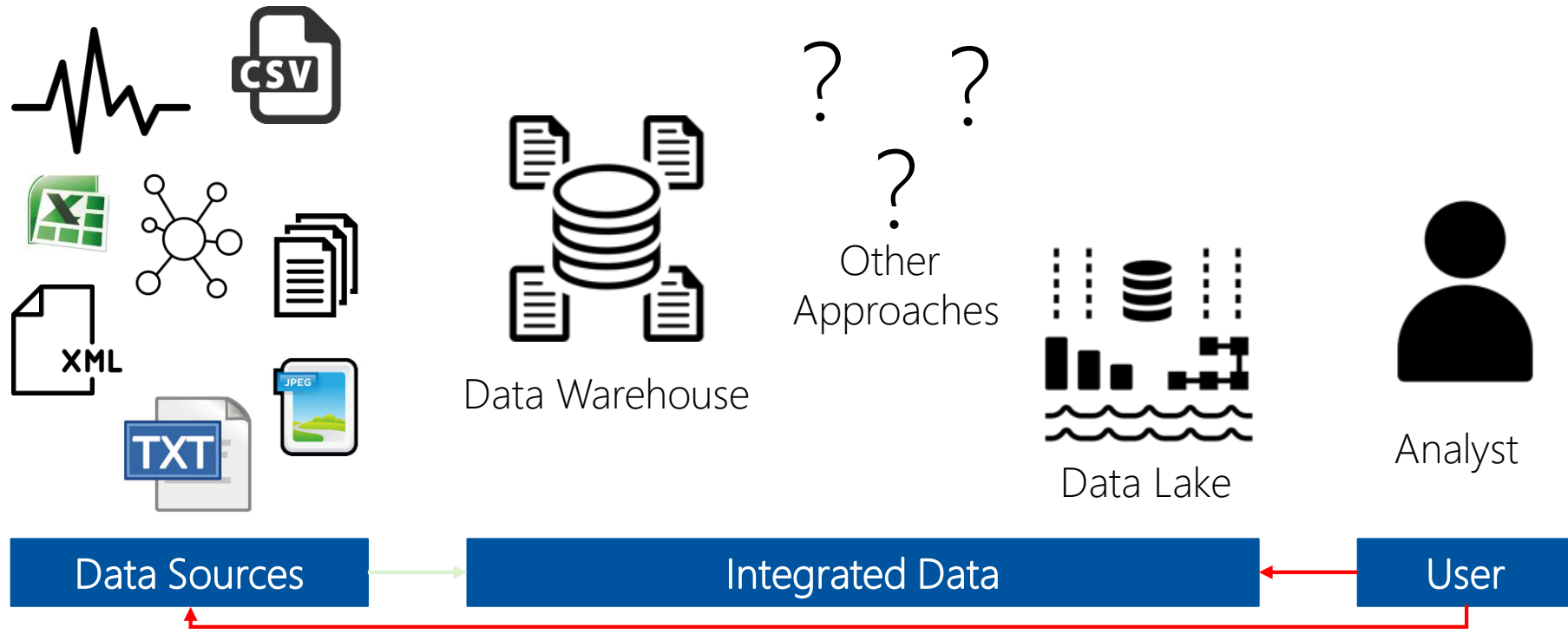
Data Pipelines & Data Integration

Why is knowledge about data access/integration important?



Your Task

... as a Data Analyst you need to know different means of **accessing** and **integrating** data from various (heterogeneous) sources.

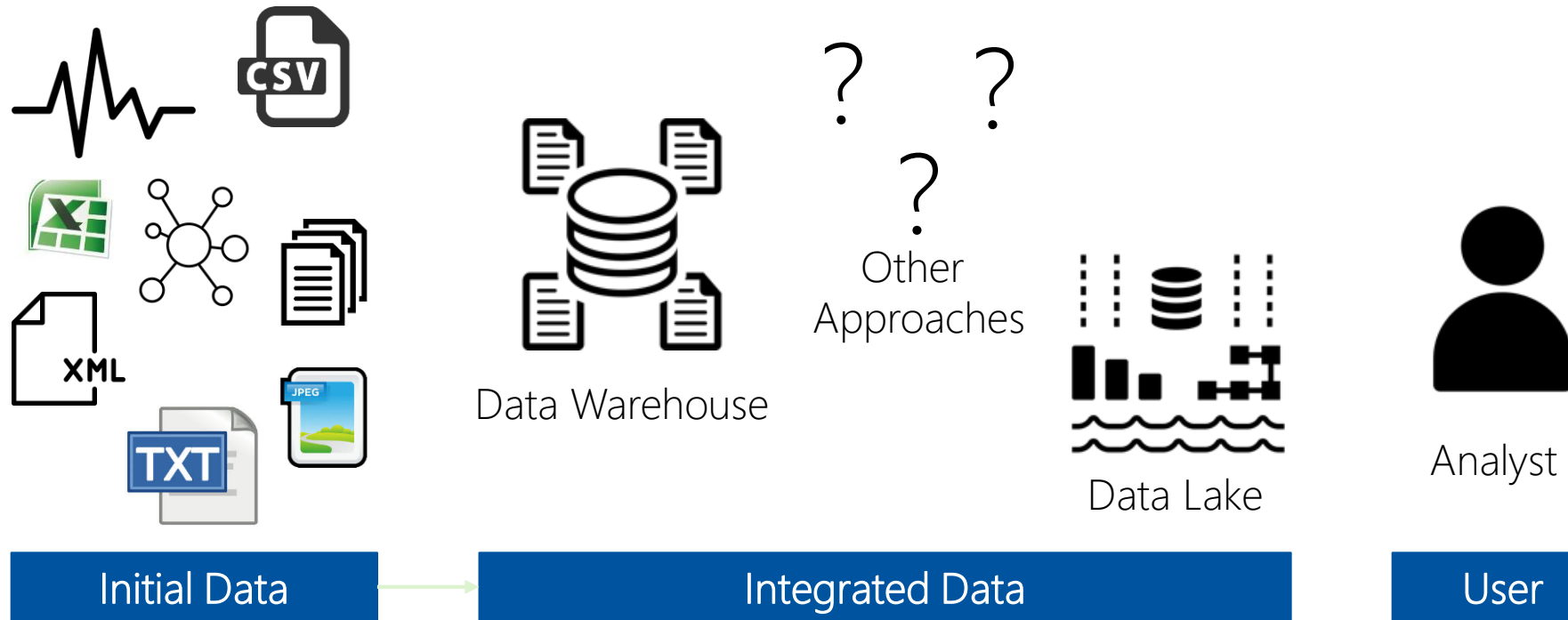


What is Data Integration?

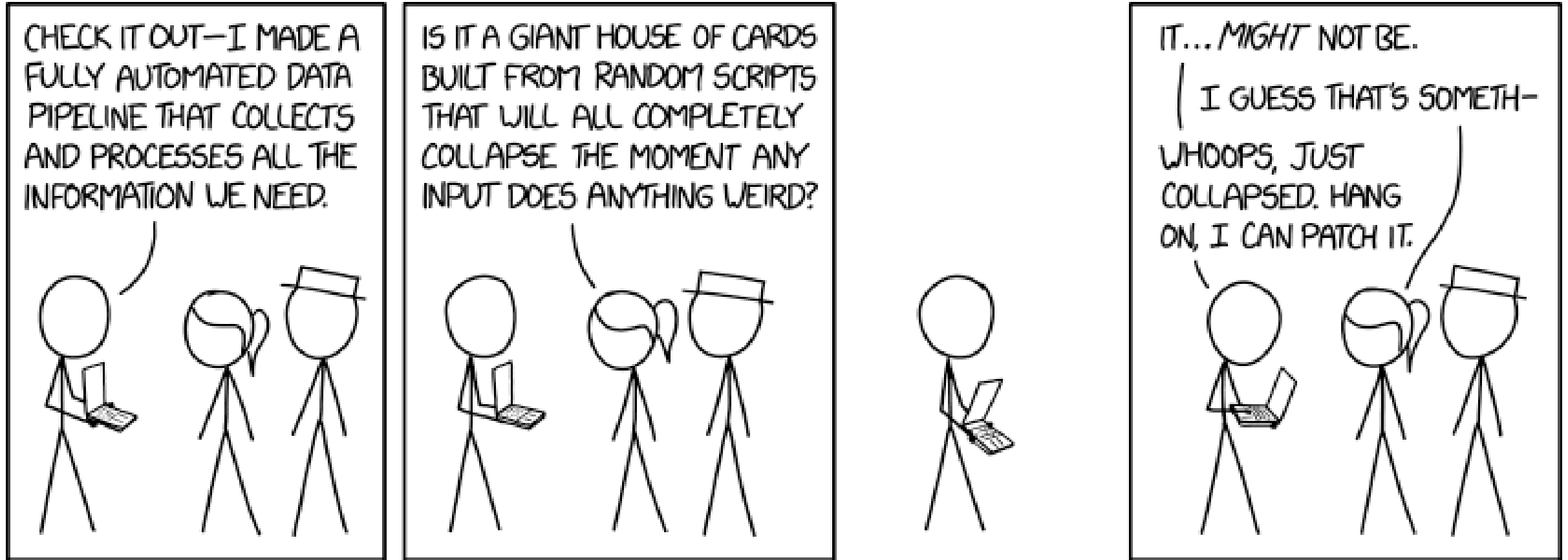


Working Definition

- Data **integration** is the task of combining different (possible heterogeneous) data from various sources to enable users a unified access to them (e.g. for data analysis).



Starting Point

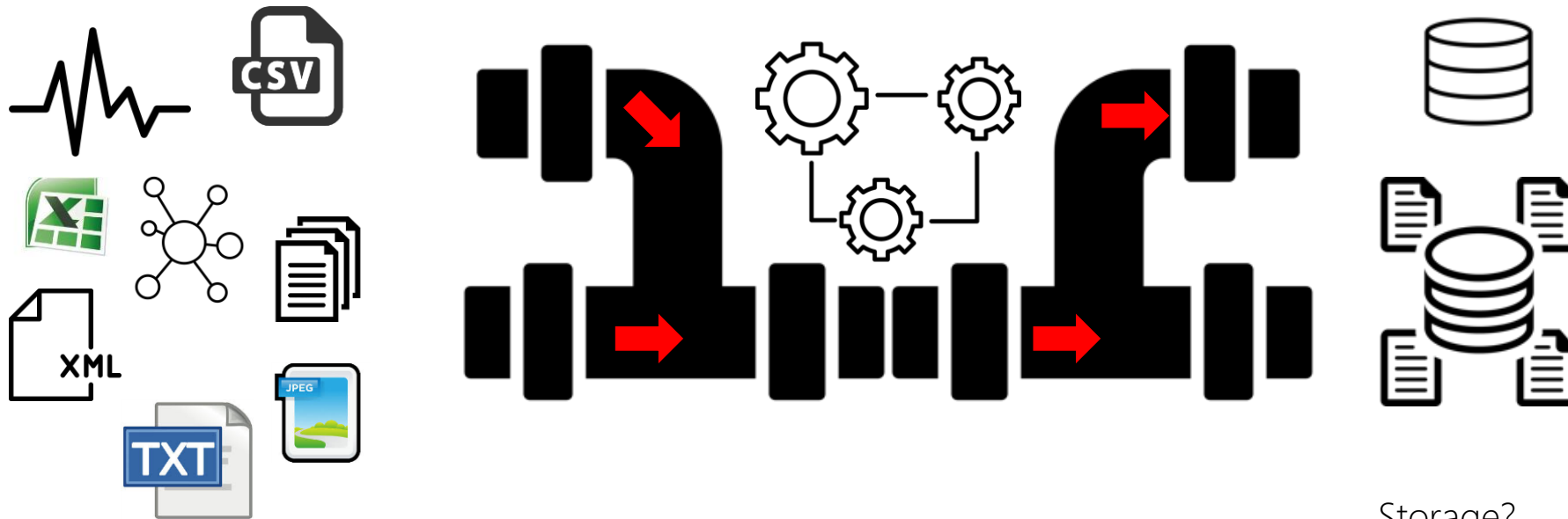


<https://xkcd.com/2054/>



Working Definition

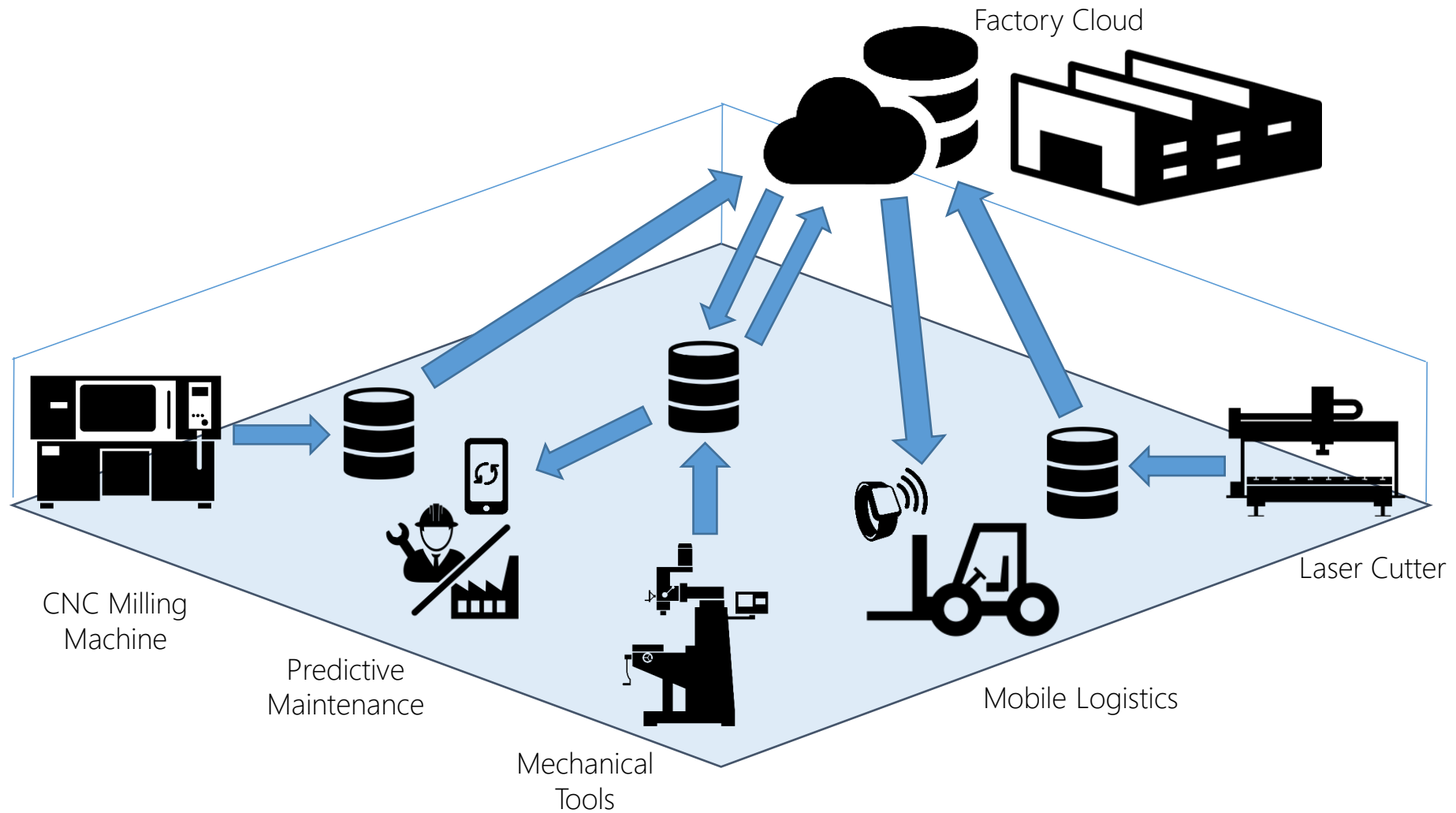
- A data pipeline is any software system that takes data from one or more inputs and transforms it in some way before writing it to one or more outputs [Tyler Akidau, Google]



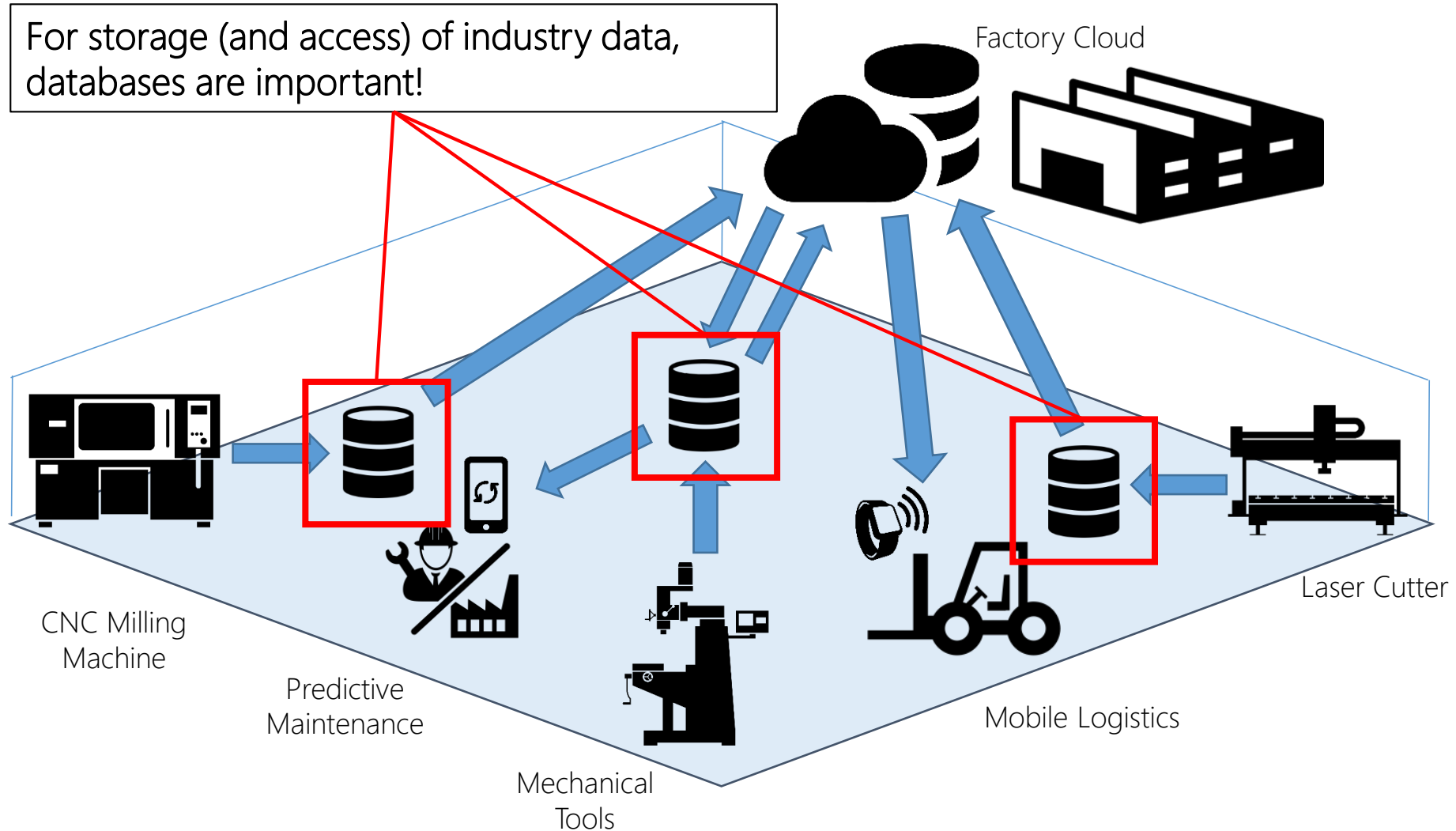
Pipelines realize integration implementation and are highly dependent on a company's IT-environment!

Storage?
Transformation?
Transportation?

A (Simple) Operational Data Pipeline



A (Simple) Operational Data Pipeline



Databases 101



What is a Database?

Databases try to separate the user/developer from the data-management effectively removing the task of organization

- Represents a **collection** of data.
- A database-management system (DBMS) provides tools to **modify/handle** the data.
- There are **different types** of representation and therefore DBMS. For example **Relational** or **Document-oriented** DBMS



Why use a Database?

Databases have several advantages compared to conventional file based systems (collections of excel-sheets, or similar loose files).

- Collections in DB are searchable and have a small footprint (resources).
- Information can be **crisscross-gathered** over the **entire** database (example queries:)
 - Which customers have capital over x amount?
 - Who lives in Aachen, owns a car and is over 25 years old?
 - What is the average income of all senior engineers over 45?
- Optimized algorithms ensure **faster search and modification**. Even multiple user support!



Why use a Database?

Databases have several advantages compared to conventional file based systems (collections of excel-sheets, or similar loose files).

- Databases are designed to be **failsafe**. This is achieved through **ACID**.
- **Easier** access for end users via **DBMS** or custom software using their **API**.
 - Custom access can (and should) prevent user errors and malicious intent.



ACID stands for

Atomicity (Transactions are either done completely or not at all)

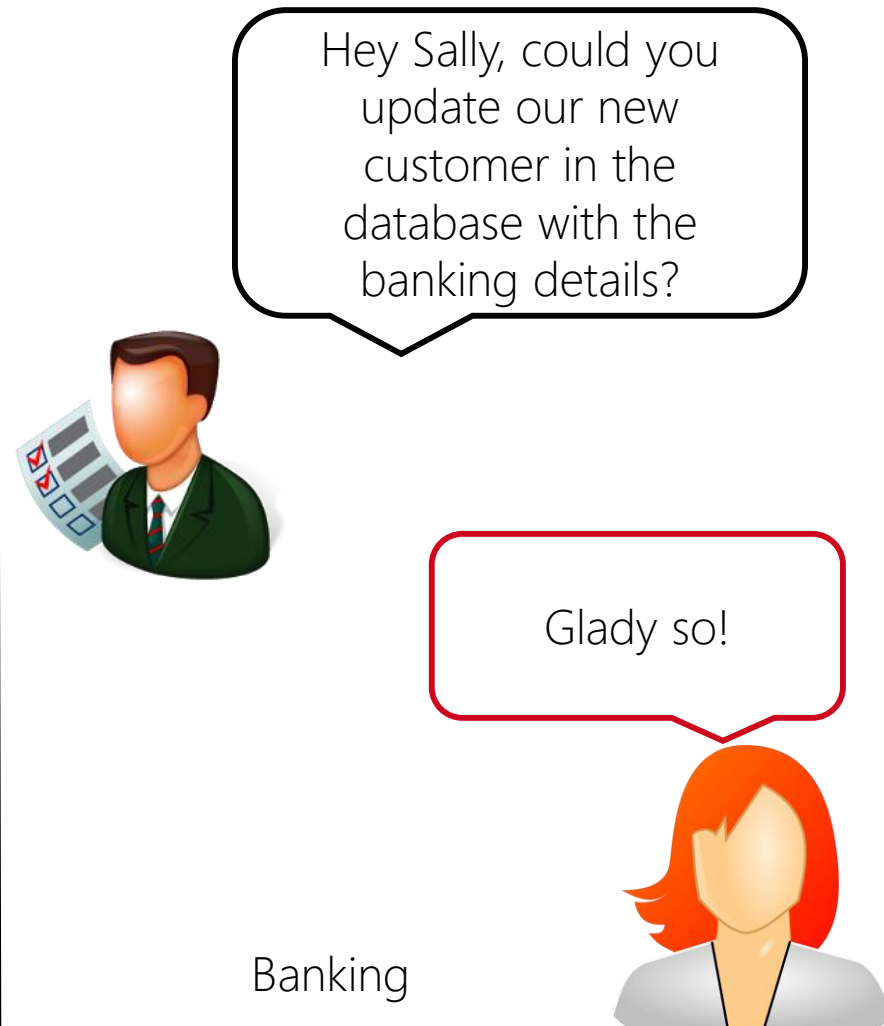
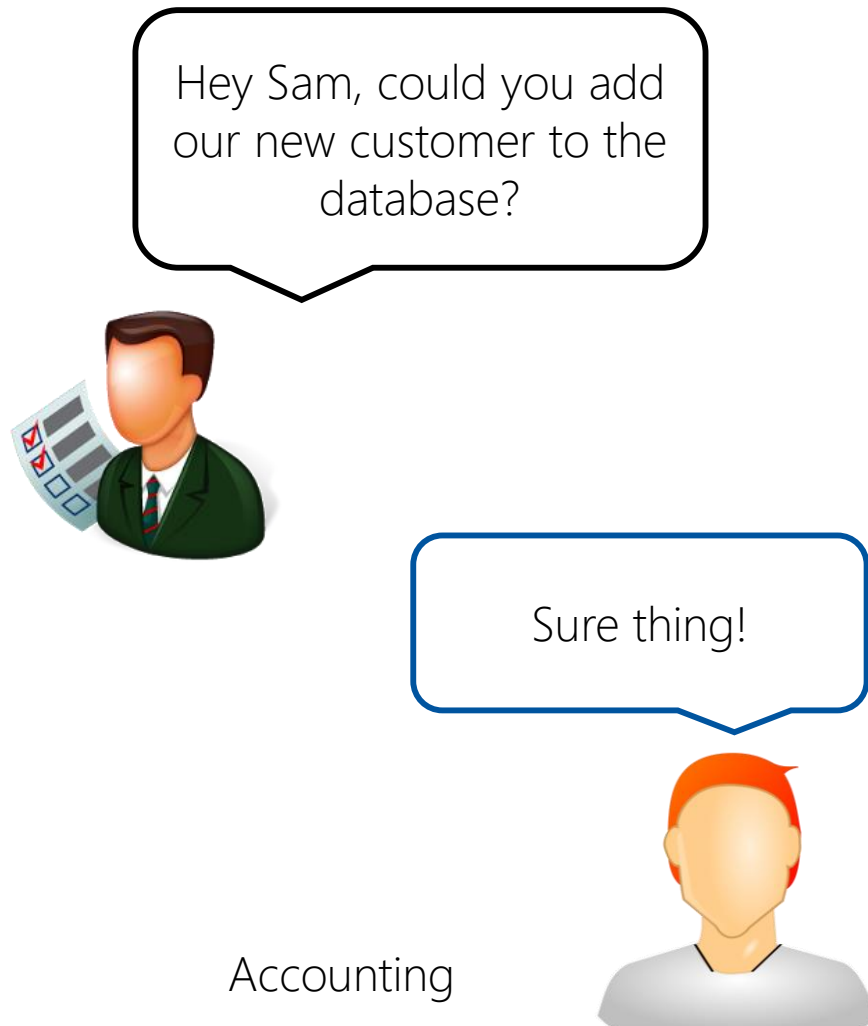
Consistency (Constraints imposed by the DB schema are always met)

Isolation (Queries don't interfere with each other)

Durability (Data is preserved in case of DB crashes, power loss etc.)

Fundamentals of Databases

Use case example



Without database

- Search for the file on hard drive.
- Try to open the file (multiuser?).
- Edit the complex/huge file without accidentally corrupting other data in the process

- Search for the file on hard drive.
- Try to open the file.
- Search user "John Doe"
- Update user "John Doe" with banking nr. and capital value.
- This is not the file that contains the banking nr.... You need to call other department.

With database

- Open database program.
- Add User "John Doe" to database.



- Open database program.
- Update user "John Doe" with banking nr. and capital value.





Relational vs Document-oriented Database?

Different approaches in database structures are used for different areas of application.

Relational DB

- Tables with columns that have a relation to other tables
- Predefined (db-)schema to guarantee conforming/fitting data

Document-oriented DB

- Documents, usually similar to JSON
- Each document can be different and complex



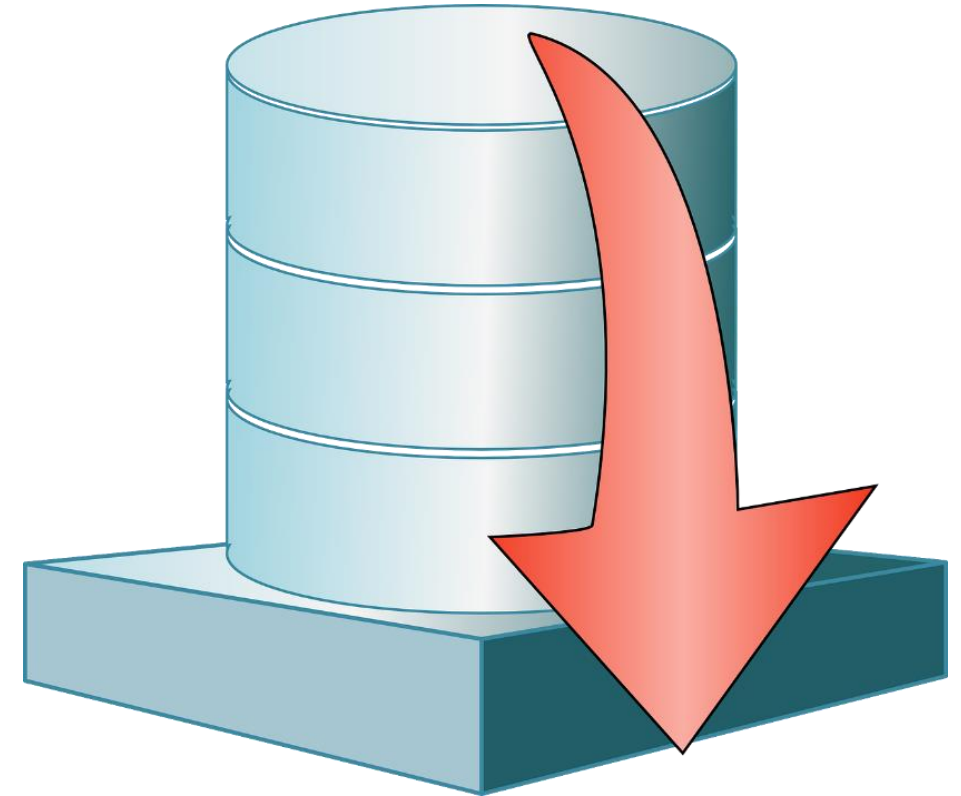
Relational vs Document-oriented Database?

Different approaches in database structures are used for different areas of application.

Relational DB	Document-oriented DB
<ul style="list-style-type: none">– Uses SQL, queries are similar to human language	<ul style="list-style-type: none">– NoSQL: Many different custom query languages
<ul style="list-style-type: none">– Not as flexible/scalable *	<ul style="list-style-type: none">– High flexibility/scalability for rapid development cycles and/or huge amounts of data (FB, Google).

Structured Query Language (SQL)

- Language for Relational Databases
 - Query Data
 - Add Data
 - Modify Data
- Based on Relational Algebra
- Close to Natural English Language



Structure of Relational Databases

- Each **Database** consists of multiple **Tables**
- Example Table: Persons

Columns

ID	LastName	FirstName	Age
1	Smith	Bill	30
2	Miller	Janine	26
3	Baker	Steven	45
4	Hunter	Anne	61

Rows

- SELECT Data

- Select all Data from Table:

SELECT * FROM TableName;

Example:

SELECT * FROM Persons;

ID	LastName	FirstName	Age
1	Smith	Bill	30
2	Miller	Janine	26
3	Baker	Steven	45
4	Hunter	Anne	61

- Select certain Columns from Table:

SELECT Column1, Column2 FROM TableName;

Example:

SELECT LastName, Age FROM Persons;

LastName	Age
Smith	30
Miller	26
Baker	45
Hunter	61

- SELECT Data with Conditions

- Select all Data from Table by Equals Condition:

SELECT * FROM TableName WHERE Condition;

Example:

SELECT * FROM Persons WHERE FirstName = 'Bill';

ID	LastName	FirstName	Age
1	Smith	Bill	30

- Select certain Data from Table by Greater Than comparison:

SELECT Column1, Column2 FROM TableName WHERE COLUMN > VALUE;

Example:

SELECT LastName, Age FROM Persons WHERE Age > 40;

LastName	Age
Baker	45
Hunter	61

■ INSERT Data

- Insert data into Table, using all columns:

INSERT INTO TableName VALUES (ValueCol1, ValueCol2,...)

Example:

INSERT INTO Persons VALUES (5, 'Tailor', 'Tom', 44)

ID	LastName	FirstName	Age
5	Tailor	Tom	44

- Insert data into Table, using certain columns

INSERT INTO TableName (Col1, Col2) VALUES (ValueCol1, ValueCol2)

Example:

INSERT INTO Persons (ID, LastName, Age) VALUES (5, 'Tailor', 44)

ID	LastName	FirstName	Age
5	Tailor	NULL	44

- UPDATE Data

- Update a value from an existing column

UPDATE TableName SET Coll = ValueColl WHERE CONDITION

Example:

UPDATE Persons SET Age = 46 WHERE ID = 3

ID	LastName	FirstName	Age
3	Baker	Steven	46

Example:

UPDATE Persons SET Age = 20 WHERE Age <= 30

ID	LastName	FirstName	Age
1	Smith	Bill	20
2	Miller	Janine	20

Relational Database

Person Table

Id (PK)	firstname	lastname	age
1	John	Smith	24

isAlive	addressid	children	spouse
true	1	null	null

Address Table

Id (PK)	street	city	state	Postal code
1	21 2nd Street	New York	NY	10021-3100

Phone Table

type	number	userid
home	212 555-1234	1
office	646 555-4567	1
mobile	123 456-7890	1

Document-oriented Database

```
{
  "firstName": "John",
  "lastName": "Smith",
  "isAlive": true,
  "age": 25,
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": "10021-3100"
  },
  "phoneNumbers": [
    {
      "type": "home",
      "number": "212 555-1234"
    },
    {
      "type": "office",
      "number": "646 555-4567"
    },
    {
      "type": "mobile",
      "number": "123 456-7890"
    }
  ],
  "children": [],
  "spouse": null
}
```

Document-oriented Database

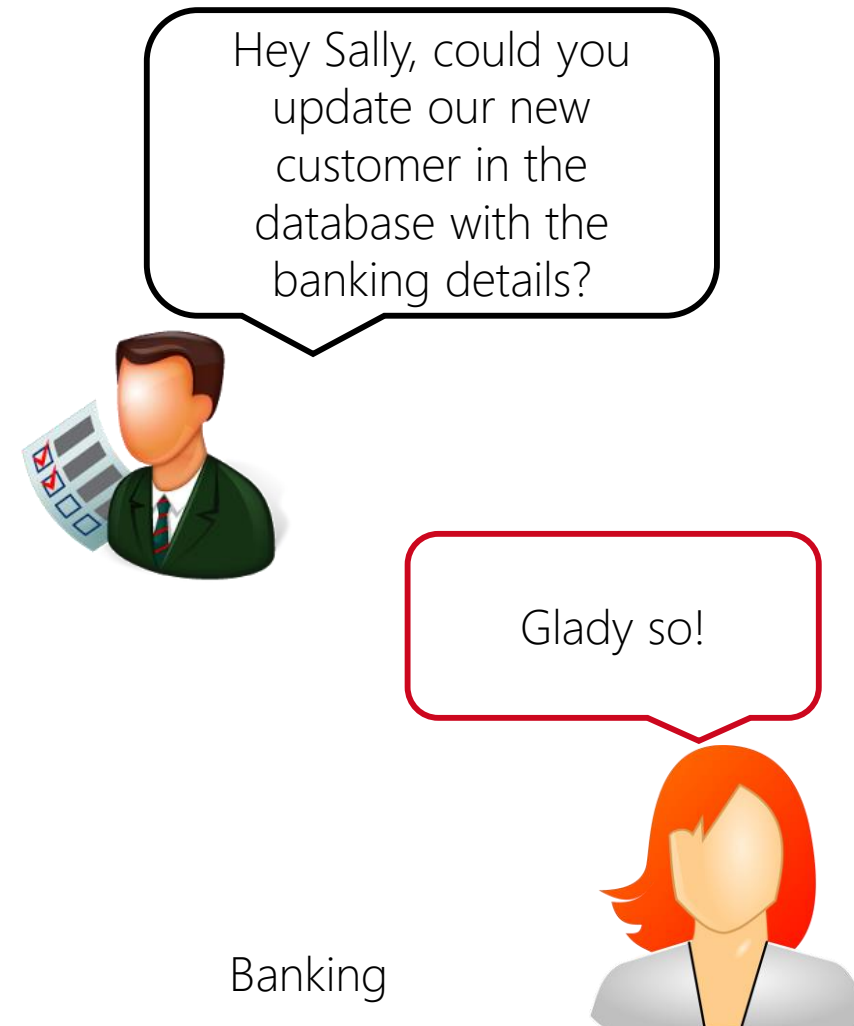
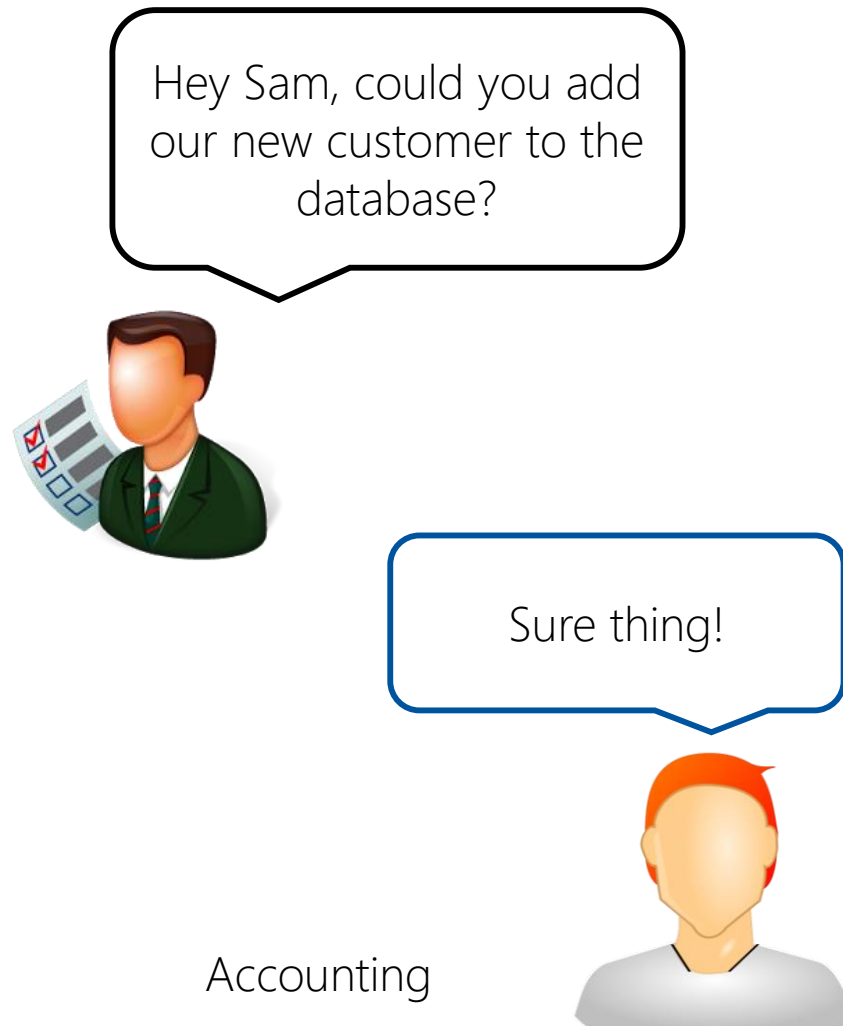
```
{
  "firstName": "John",
  "lastName": "Smith",
  "isAlive": true,
  "age": 25,
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": "10021-3100"
  },
  "phoneNumbers": [
    {
      "type": "home",
      "number": "212 555-1234"
    },
    {
      "type": "office",
      "number": "646 555-4567"
    },
    {
      "type": "mobile",
      "number": "123 456-7890"
    }
  ],
  "children": [],
  "spouse": null
}
```

JSON to MongoDB Document

```
{
  "_id" : ObjectId("59a2cd843868cb491e526966"),
  "firstName" : "John",
  "lastName" : "Smith",
  "isAlive" : true,
  "age" : 25,
  "address" : {
    "streetAddress" : "21 2nd Street",
    "city" : "New York",
    "state" : "NY",
    "postalCode" : "10021-3100"
  },
  "phoneNumbers" : [
    {
      "type" : "home",
      "number" : "212 555-1234"
    },
    {
      "type" : "office",
      "number" : "646 555-4567"
    },
    {
      "type" : "mobile",
      "number" : "123 456-7890"
    }
  ],
  "children" : [ ],
  "spouse" : null
}
```

Fundamentals of Databases

Use case example (concrete)

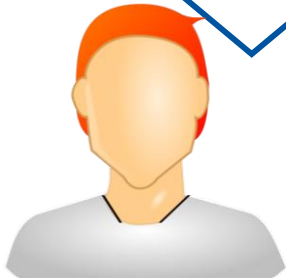


Fundamentals of Databases

Use case example (concrete)

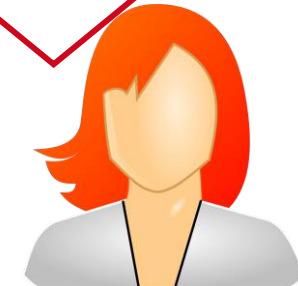
Add user "John Doe" to database:

```
db.collection.insert(  
  {  
    "name": "john",  
    "last_name": "doe"  
  })
```



Update user "John Doe":

```
db.collection.findOneAndUpdate(  
  {  
    "name": "john",  
    "last_name": "doe"  
  }, {  
    $set: {  
      "banknr": "12345",  
      "capital": 50000  
    }  
  })
```



Fundamentals of Databases

Use case example (concrete)

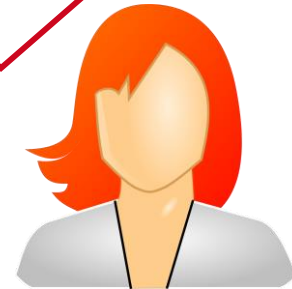
Hey Sally, could you tell me the amount of customers with 50000 or more capital?



I'm on it!



```
db.collection.aggregate([
  { $match:
    { "capital":
      { $gte: 50000 }
    }
  },
  { $group:
    { _id: null, "count":
      { $sum: 1 }
    }
  }
] );
```



Approaches to Data Integration

Extract, Transform, Load: The ETL Process



Working Definition

- ETL is a process in which data is first extracted (E) from various data sources, then transformed (T), e.g. cleaned or schema adjusted, and finally loaded (L) into the target system, e.g. a data warehouse.

Extract

- Various (heterogeneous data sources)
- Synchronously
- Asynchronously
 - Periodically
 - Event-based
 - Query-based

Transform

- Uniform schema (Schema-Mapping)
- Syntactic Transform
- Semantic Transform, e.g. duplicate removal
- Data Preprocessing, Cleansing (see lecture 3)

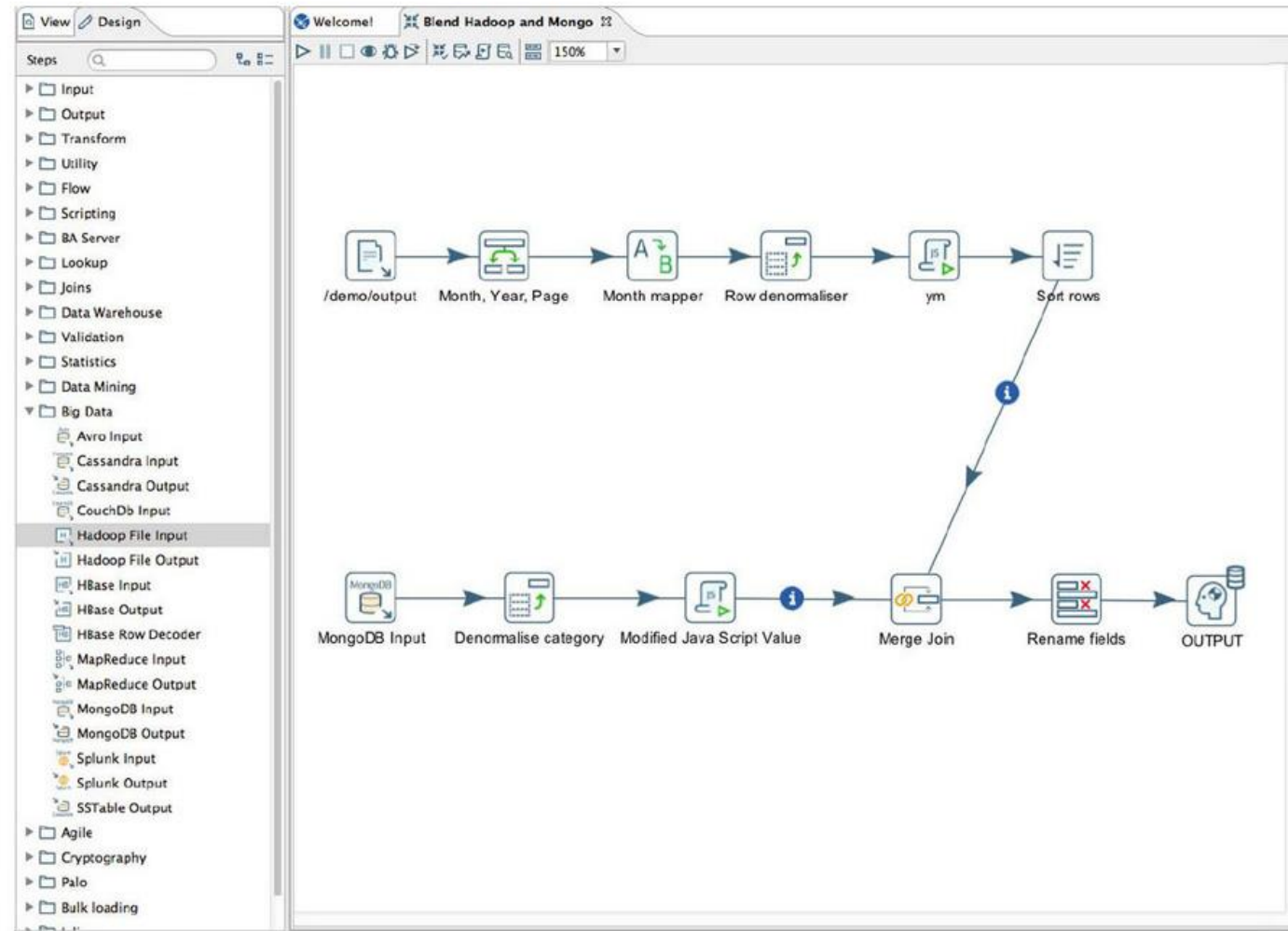
Load

- Overwrite
- Update
- Versioning

Companies



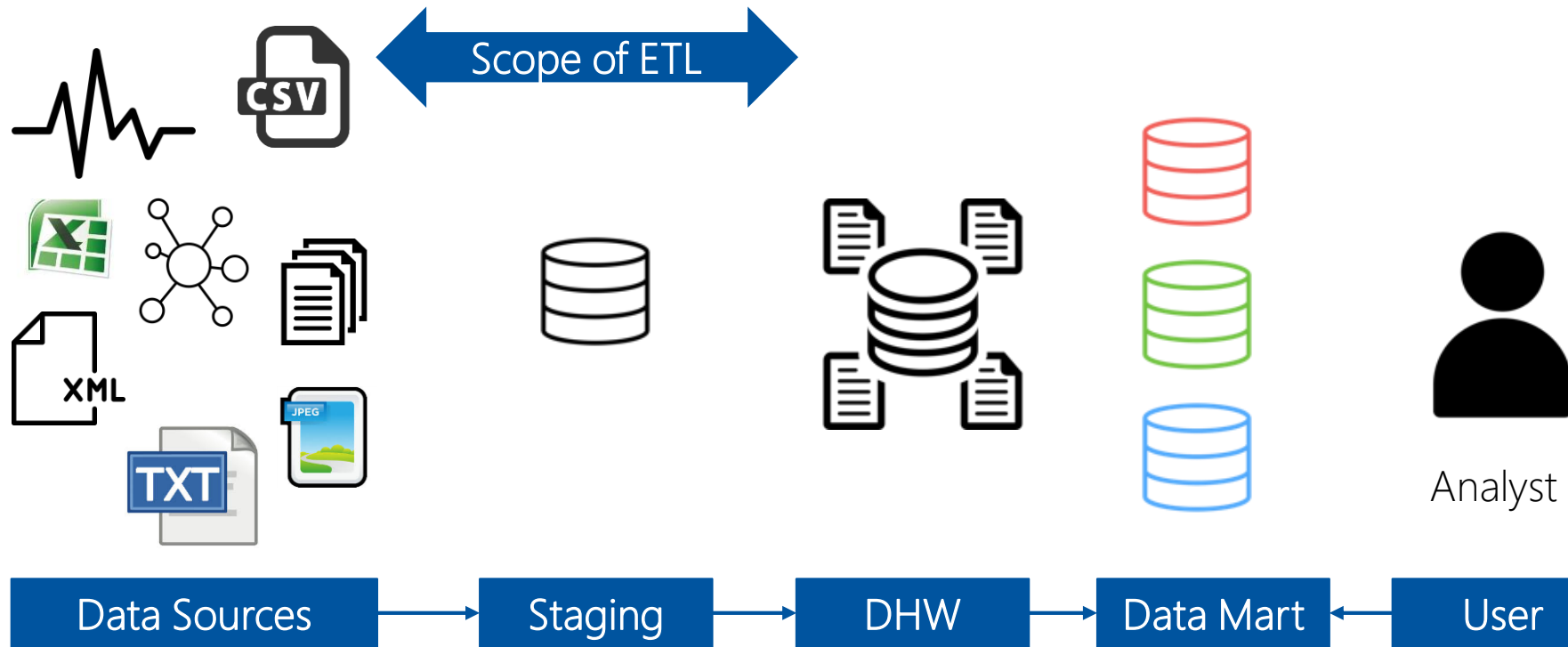
Extract, Transform, Load: The ETL Process



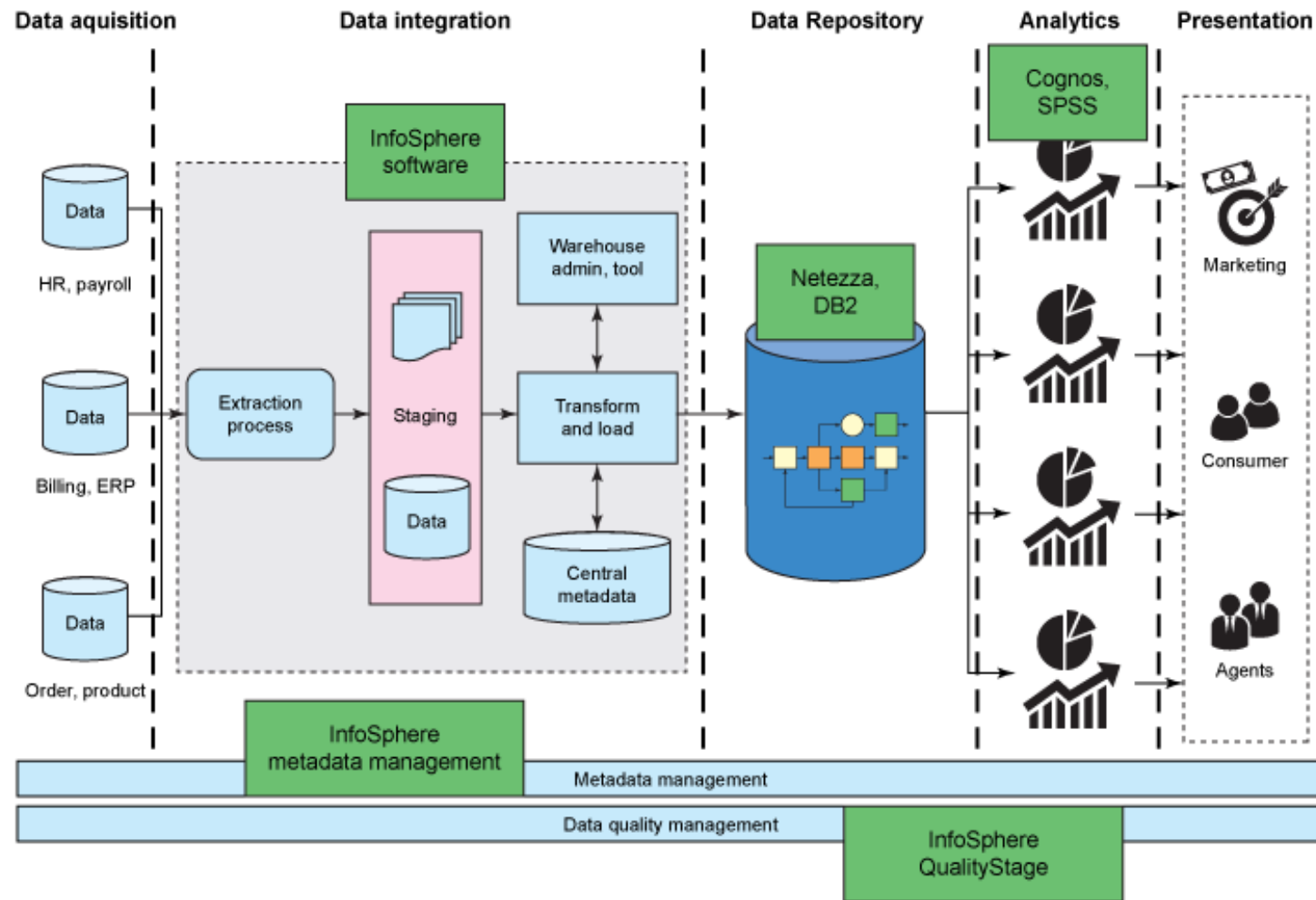


Working Definition

- A **data warehouse** (DWH) is a central database system which integrates data from all kinds of company-wide operational data sources, e.g. production, for subsequent analysis purposes.



Data Warehouse

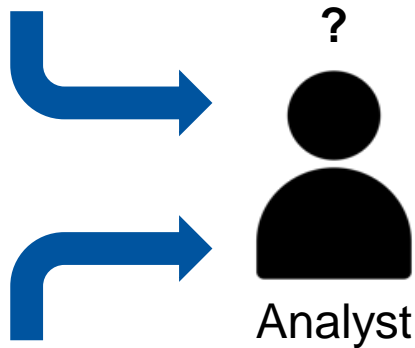


<https://developer.ibm.com/tutorials/ba-augment-data-warehouse1/>



Schema-on-write

- For a data warehouse one has to define the schema (e.g. columns, datatypes, relationships etc.) of the database before any data is written into it.



- Data has been altered before the analysis
- Specific structure for a specific purpose, e.g. accounting analysis
- Time-consuming to adapt when analysis purposes changes



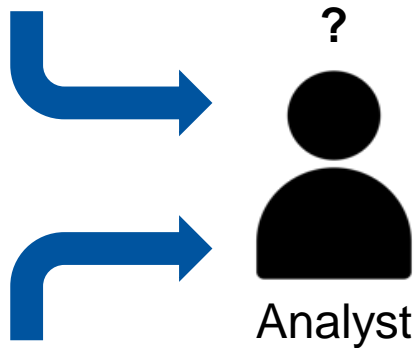
ETL and Data Warehouses

- For a data warehouse one has to define the transform step of ETL. That is, how the data is modified before it is persisted and before it is first accessed by any analyst.



Schema-on-Read

- With a schema-on-read approach data is not structured/transformed while it is persisted, but only when it is read out of the storage. Furthermore the T step of ETL is delegated to the actual data user.



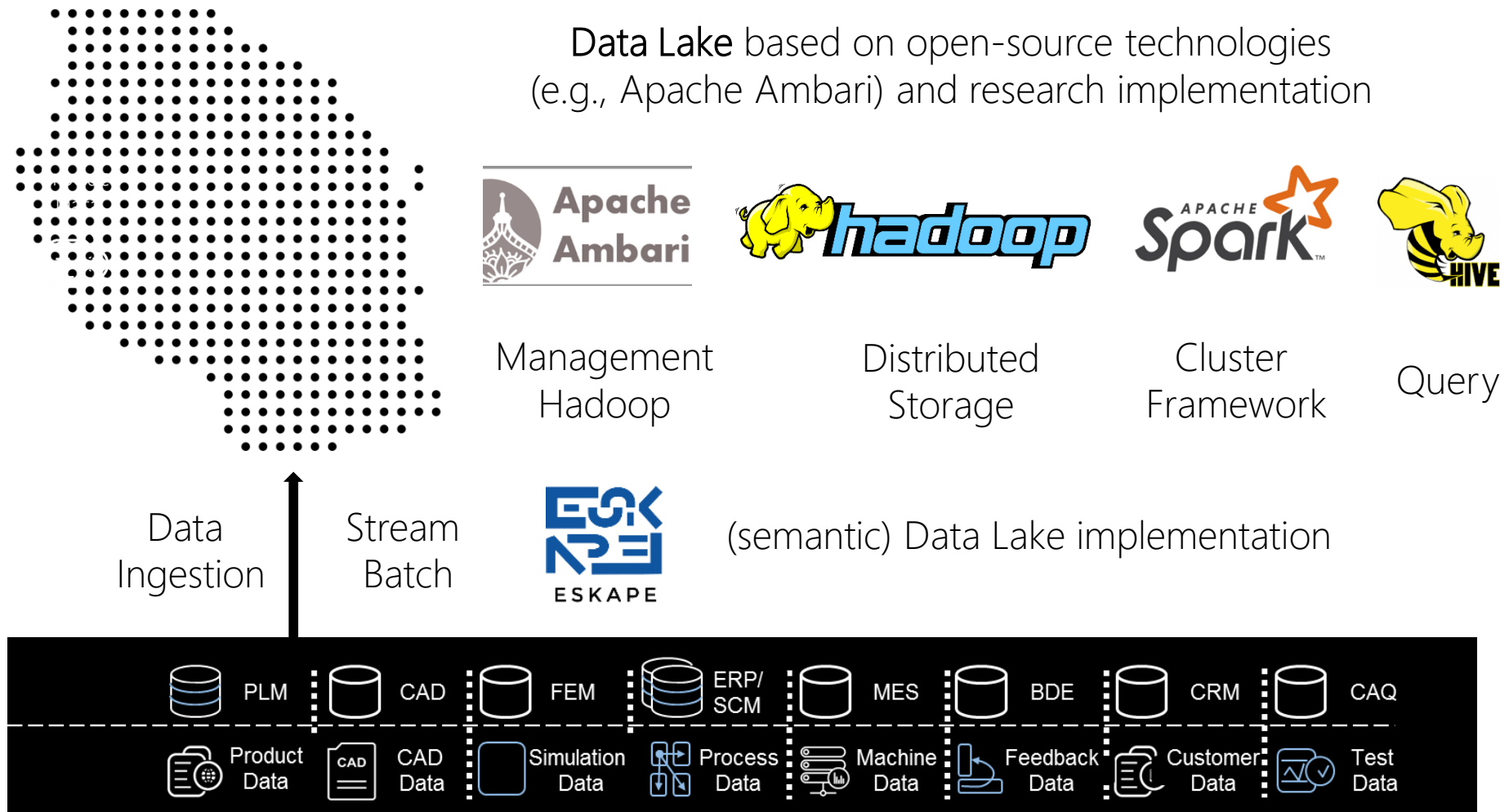
- Data has is not altered before the analysis
- **Nothing is discarded before storage**
- There is no fixed purposed imposed by the schema of the storage (read: there is no schema)
- Challenge: Finding data!



Data Lakes

- A data lake is a central repository for storing data in its natural format, i.e. it includes raw copies of the data from the data sources. Data lakes realize the schema-on-read approach

(One) Data Lake Technology Stack



Data Warehouse vs Data Lakes

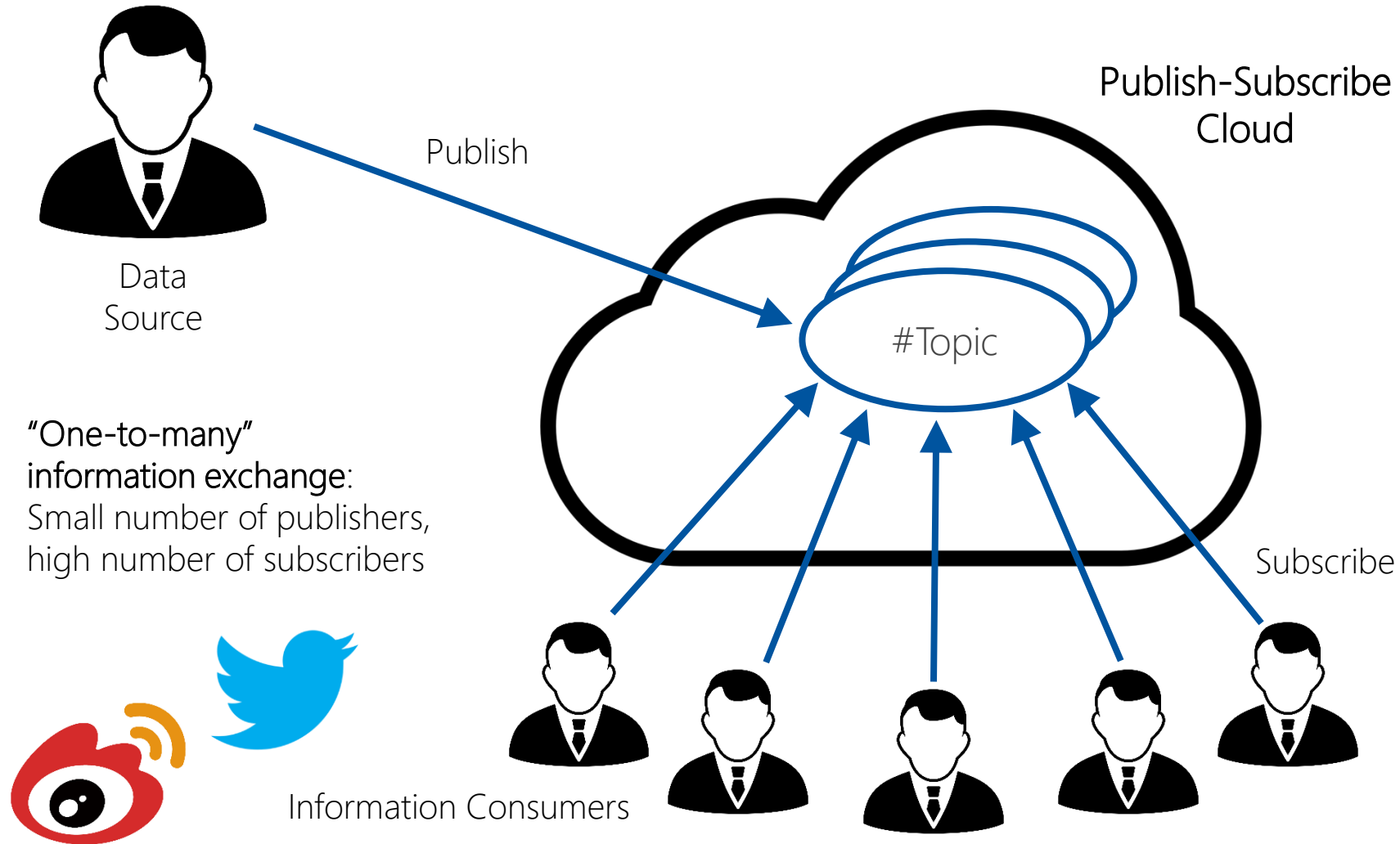
Characteristics	Data Warehouse	Data Lake
Data	Relational from transactional systems, operational databases, and line of business applications	Non-relational and relational from IoT devices, web sites, mobile apps, social media, and corporate applications
Schema	Designed prior to the DW implementation (schema-on-write)	Written at the time of analysis (schema-on-read)
Price/Performance	Fastest query results using higher cost storage	Query results getting faster using low-cost storage
Data Quality	Highly curated data that serves as the central version of the truth	Any data that may or may not be curated (ie. raw data)
Users	Business analysts	Data scientists, Data developers, and Business analysts (using curated data)
Analytics	Batch reporting, BI and visualizations	Machine Learning, Predictive analytics, data discovery and profiling

<https://aws.amazon.com/de/big-data/datalakes-and-analytics/what-is-a-data-lake/>

Excursus: Transporting the Data

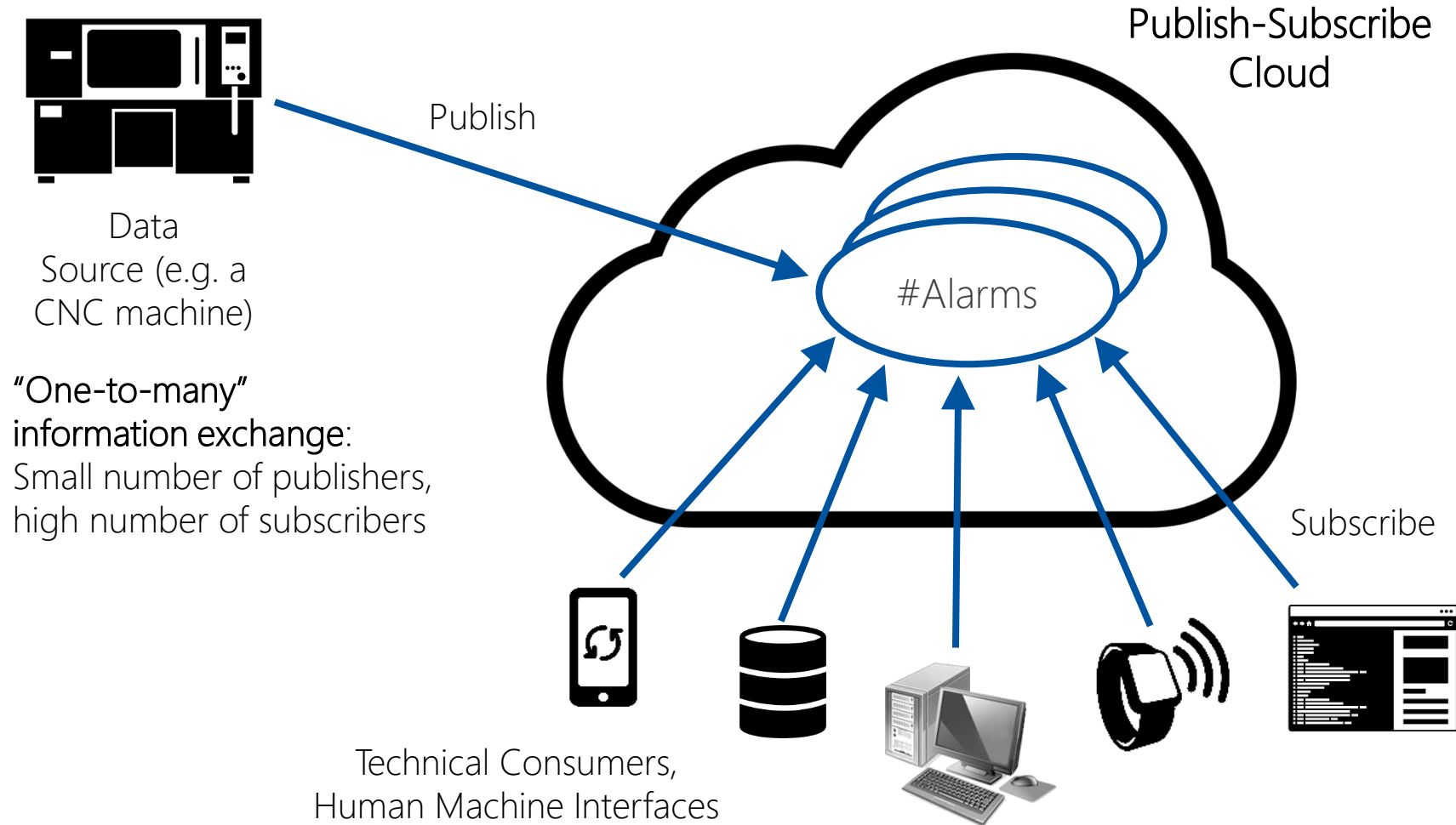
Internet of Production

Publish/Subscribe Mechanisms – Twitter Like Communication



Internet of Production

Publish/Subscribe Mechanisms – Manufacturing Environment



Message Queue Telemetry Transport (MQTT) is an easy-to-use publish/subscribe real-time protocol for lightweight systems

- MQTT is an **M2M protocol** with minimum overhead that was developed especially for SCADA (Supervisory Control and Data Acquisition) purposes
- MQTT is “data agnostic” – The protocol transfers any kind of information, from text to **binary** encoded content
- MQTT is based on **TCP** and can be easily encrypted with SSL/TLS
- MQTT is one of the most-promising **Internet of Things (IoT) standard** as it is implementable in almost all devices and can be used in various applications
- MQTT uses a **broker** for addressing published messages on the right channels
- MQTT enables **real-time push**, i.e. messages are delivered immediately after being publishes. MQTT does not support polling though

Obermeier, D. (2015): MQTT: Schnelleinstieg in das schlanke IoT-Protokoll mit Java. IoT-Allrounder. Hg. v. jaxenter. Online available at <https://jaxenter.de/iot-allrounder-27208>, last accessed on 06.08.2016

- General Structure of MQTT message:

```
message = {  
    'topic': 'factory/drilling/machine1/tempSensor2',  
    'payload':  
        {  
            'value': 28.48,  
            'unit': 'Celsius',  
            'sensor_type': 'DHT11'  
            ...  
        }  
}
```

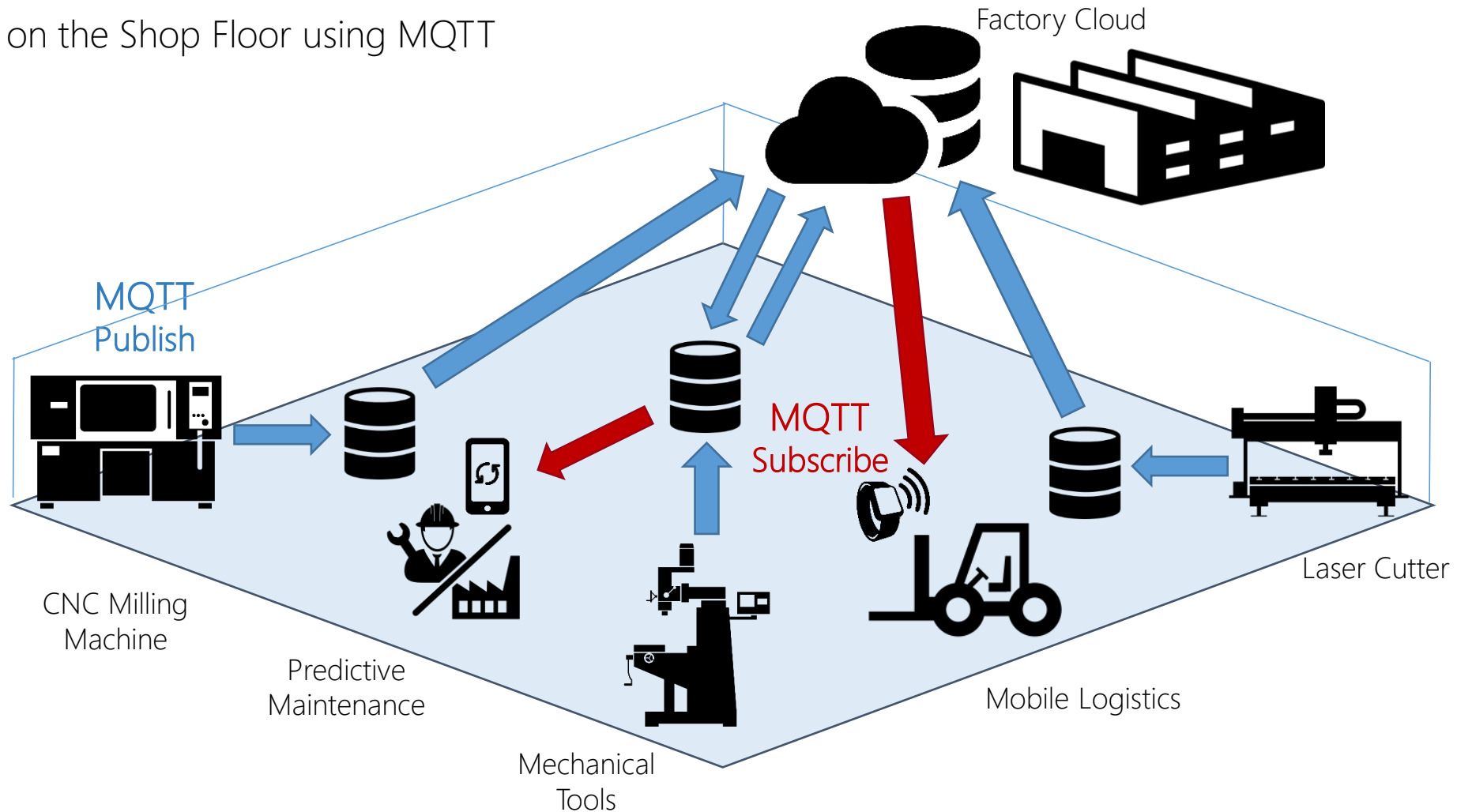
- Topics can be used to add meta-data, e.g. location:

```
'factory/drilling/machine1/tempSensor1'  
'factory/drilling/machine1/tempSensor2'  
'factory/drilling/machine1/pressureSensor1'  
'factory/milling/machine1/tempSensor1'  
'factory/milling/machine2/tempSensor1'
```

```
factory  
-- drilling  
    -- machine1  
        -- tempSensor1  
        -- tempSensor2  
        -- pressureSensor1  
-- milling  
    -- machine1  
        -- tempSensor1  
    -- machine2  
        -- tempSensor1
```

Turning Mechanical Processes into Knowledge

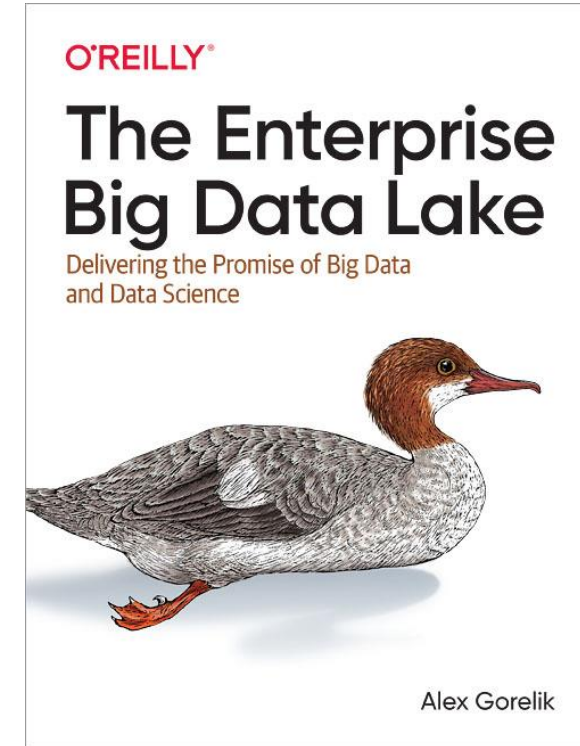
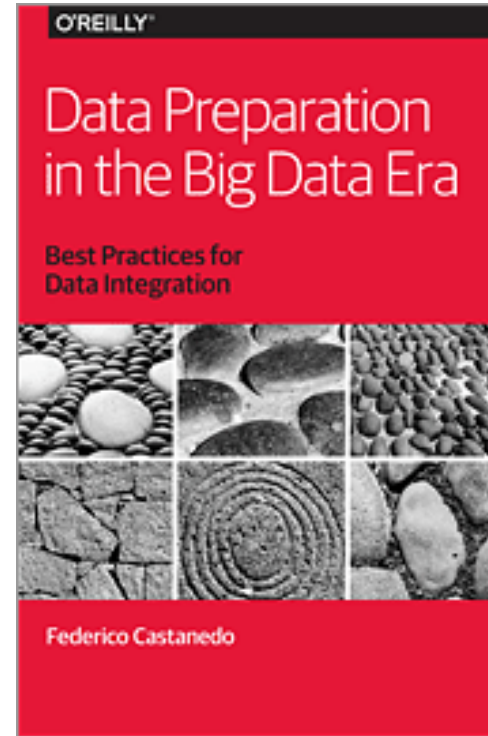
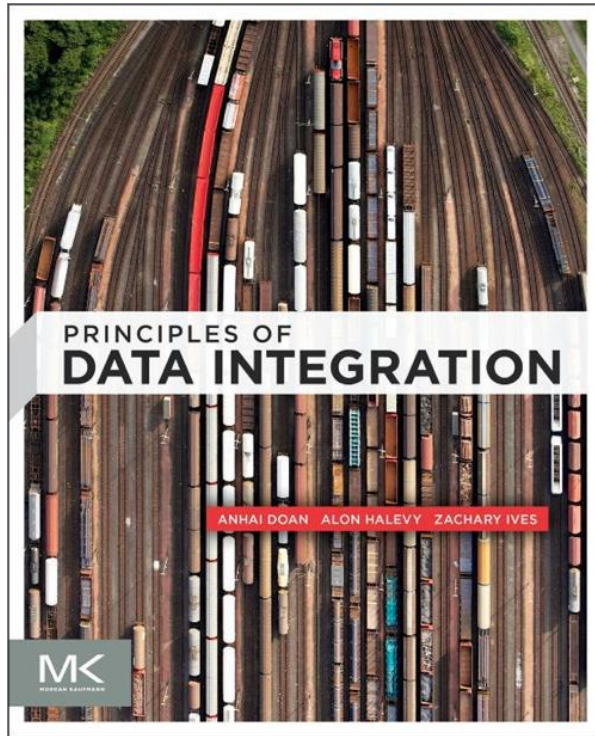
- Connectivity on the Shop Floor using MQTT



Further Reading Material

<https://github.com/pentaho/pentaho-kettle>

<https://www.youtube.com/watch?v=v5lkNHib7bw> (AWS re:Invent 2018: Effective Data Lakes: Challenges and Design Patterns)



Thank you for your attention!

Lecture Team AIDAE

aidae@ima-ifu.rwth-aachen.de