PHUSE EU Connect 2022

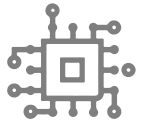
*Device and Decentralized Clinical Trials using a Storage and **A**dvanced Ana**ly**tics Platform for the **C**linical Data **E**nvironment (ALYCE)*

Presentation #: DH09

Stream: Data Handling & Data Engineering

Presenter: Stephan Cichos (Bayer AG)

Agenda



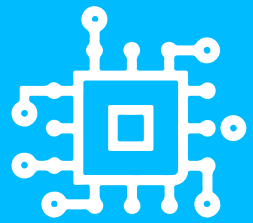
Current #DataEngineering Challenges

„Softlanding“ New Data Sources | Our Approach

Summary

PHUSE EU Connect 2022

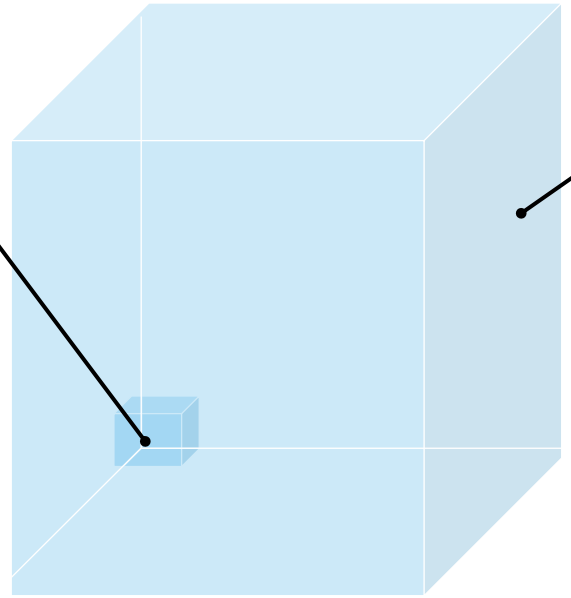
Current #DataEngineering Challenges



Adding new data sources – what will change?

Data collected at on-site visits

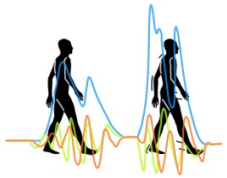
- // Demographics
- // Weight, Height
- // Vital Signs
- // Single 12 Lead ECGs
- // Adverse Events, Medical History
- // …



Data from Digital Health Technologies collected at optimal time points e.g.:

Actigraphy devices

- // Physical Activity
- // Energy Expenditure
- // Sleep Activity
- // Body Posture

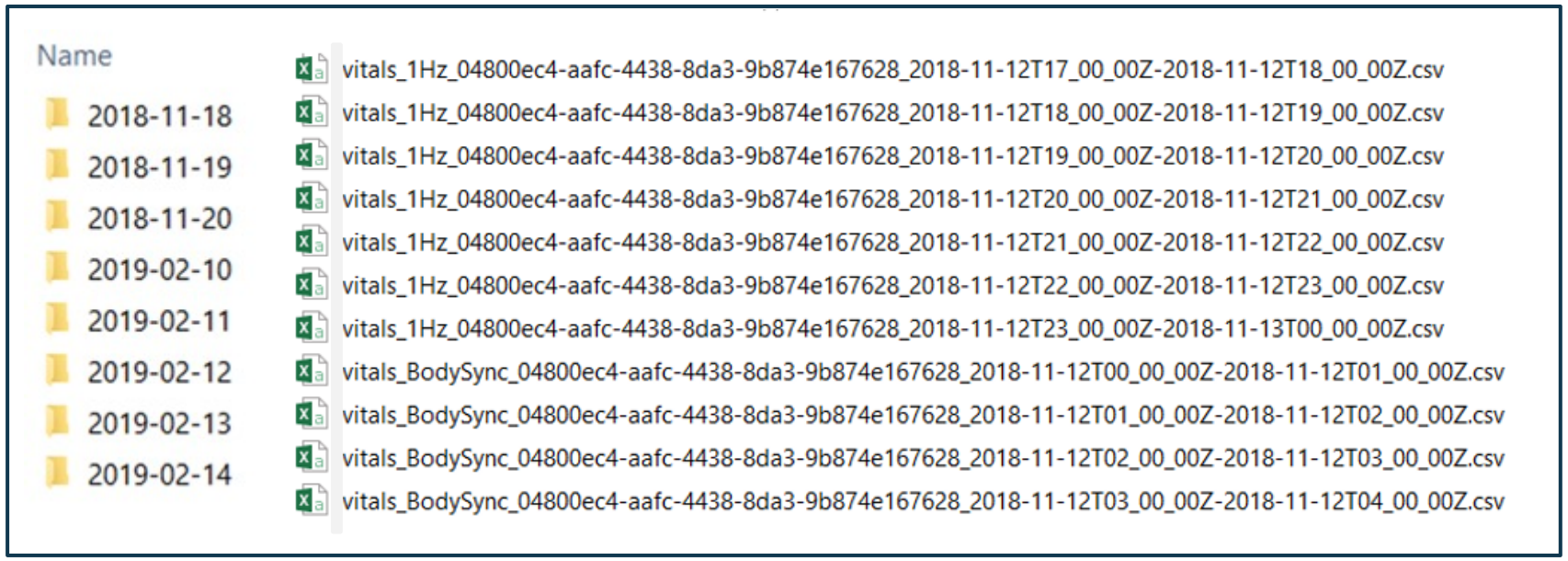


ECG Patches

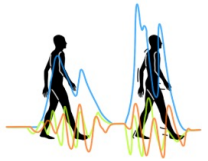
- // Heart Rate
- // Respiratory Rate
- // R-to-R Interval
- // Fluid Status

Volume, Speed and Complexity of generated data will significantly increase

Example - ECG and Activity data



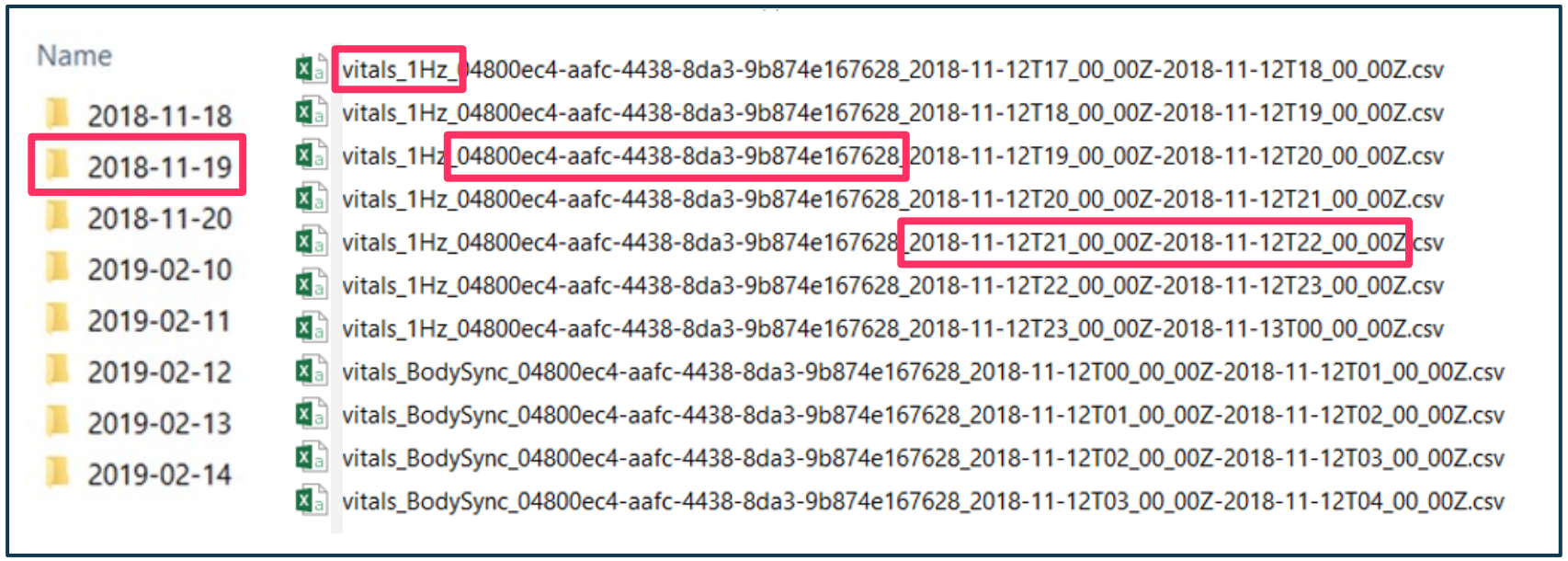
- Heart Rate
- Respiratory Rate
- R-to-R interval
- Fluid Status…



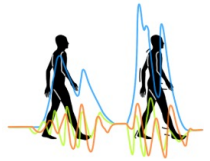
- Physical Activity
- Sleep Activity
- …

- ~300,000 datafiles spread over hundreds of folders (1.6 TB in total) for 80 patients
- 7 datafiles per category per hour (not more than 1 - 6 variables per file)
- Unique Patient- and Device identifier as part of the filename
- Timestamps are in UTC collected and need to get converted (by e.g. using IANA Timezone DB)

Example - ECG and Activity data



- Heart Rate
- Respiratory Rate
- R-to-R interval
- Fluid Status…



- Physical Activity
- Sleep Activity
- …

- ~300,000 datafiles spread over hundreds of folders (1.6 TB in total) for 80 patients
- 7 datafiles per category per hour (not more than 1 - 6 variables per file)
- Unique Patient- and Device identifier as part of the filename
- Timestamps are in UTC collected and need to get converted (by e.g. using IANA Timezone DB)

Example - ECG data in high resolution

	heart_rate	memory	respiration_rate	skin_temp	battery	stream	applied_status
2019-12-25 04:00:01.922000+00:00	78.0	0.0	18.0	33.0	67.0	live_data	applied
2019-12-25 04:00:05.922000+00:00	79.0	0.0	18.0	33.0	67.0	live_data	applied
2019-12-25 04:00:09.922000+00:00	80.0	0.0	18.0	33.0	67.0	live_data	applied
2019-12-25 04:00:13.922000+00:00	78.0	0.0	18.0	33.0	67.0	live_data	applied
2019-12-25 04:00:17.922000+00:00	77.0	0.0	18.0	33.0	67.0	live_data	applied
2019-12-25 04:00:21.922000+00:00	79.0	0.0	17.0	33.0	67.0	live_data	applied
2019-12-25 04:00:25.922000+00:00	81.0	0.0	17.0	33.0	67.0	live_data	applied
2019-12-25 04:00:29.922000+00:00	80.0	0.0	17.0	33.0	67.0	live_data	applied
2019-12-25 04:00:33.922000+00:00	79.0	0.0	17.0	33.0	67.0	live_data	applied
2019-12-25 04:00:37.922000+00:00	77.0	0.0	16.0	33.0	67.0	live_data	applied
2019-12-25 04:00:41.922000+00:00	80.0	0.0	16.0	33.0	67.0	live_data	applied
2019-12-25 04:00:45.922000+00:00	78.0	0.0	16.0	33.0	67.0	live_data	applied

// Not every data point from the device need to become part of the operational database

// Overhead in the Data Warehouse → Inefficient

// Data can be unclean (redundant, inconsistent, missing data, extreme outliers/artefacts)

Example - ECG data in high resolution

	heart_rate	memory	respiration_rate	skin_temp	battery	stream	applied_status
2019-12-25 04:00:01.922000+00:00	78.0	0.0	18.0	33.0	67.0	live_data	applied
2019-12-25 04:00:05.922000+00:00	79.0	0.0	18.0	33.0	67.0	live_data	applied
2019-12-25 04:00:09.922000+00:00	80.0	0.0	18.0	33.0	67.0	live_data	applied
2019-12-25 04:00:13.922000+00:00	78.0	0.0	18.0	33.0	67.0	live_data	applied
2019-12-25 04:00:17.922000+00:00	77.0	0.0	18.0	33.0	67.0	live_data	applied
2019-12-25 04:00:21.922000+00:00	79.0	0.0	17.0	33.0	67.0	live_data	applied
2019-12-25 04:00:25.922000+00:00	81.0	0.0	17.0	33.0	67.0	live_data	applied
2019-12-25 04:00:29.922000+00:00	80.0	0.0	17.0	33.0	67.0	live_data	applied
2019-12-25 04:00:33.922000+00:00	79.0	0.0	17.0	33.0	67.0	live_data	applied
2019-12-25 04:00:37.922000+00:00	77.0	0.0	16.0	33.0	67.0	live_data	applied
2019-12-25 04:00:41.922000+00:00	80.0	0.0	16.0	33.0	67.0	live_data	applied
2019-12-25 04:00:45.922000+00:00	78.0	0.0	16.0	33.0	67.0	live_data	applied

// Not every data point from the device need to become part of the operational database

// Overhead in the Data Warehouse → Inefficient

// Data can be unclean (redundant, inconsistent, missing data, extreme outliers/artefacts)

Hundreds of Parameters & Gigabytes of Data

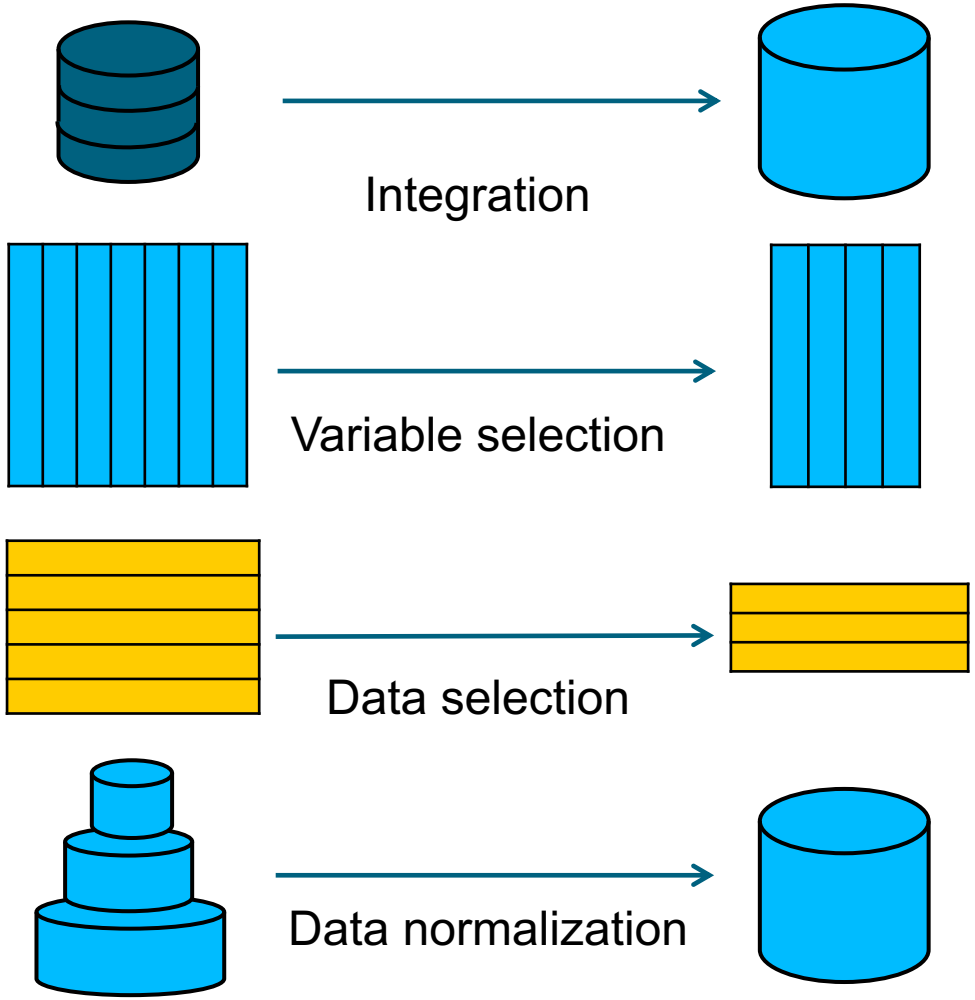
[Source: https://i.pinimg.com/originals/67/f9/f4/67f9f436e3b2ac3edb8d212edcaf4d75.jpg]

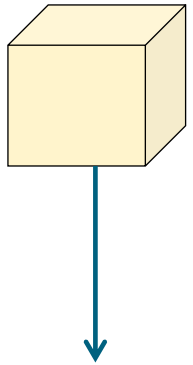
Capabilities / Requirements / Expectations to be discussed with the vendor very early in the process during set-up

Vendor

- Vendors often cannot provide data as per sponsor data model
- Vendors device data model often more efficient than SDTM (e.g. horizontal vs. vertical data structure)
- Additional Data Aggregations may be required later on (e.g. from minutes to daily resolution)
- For large amount of data, new data transfer methods need to be explored

Generated data requires additional processing / wrangling





- Transformation to e.g. SDTM
- Push to operational database(s)
- Push to statistical analysis environment

STUDYID	SUBJIDN	TEST	ORRES	ORRESU	DRVFL	_D	_M	_Y	_TM	EN_D	EN_M	EN_Y	EN_TM
42109	99990001	HRMAX15	75	beats/min	Y	2	11	2019	10:00	2	11	2019	10:15
42109	99990001	HRMIN15	72	beats/min	Y	2	11	2019	10:00	2	11	2019	10:15
42109	99990001	HRMN15	73	beats/min	Y	2	11	2019	10:00	2	11	2019	10:15
42109	99990001	RESPMAX15	18	breaths/min	Y	2	11	2019	10:00	2	11	2019	10:15
42109	99990001	RESPMIN15	15	breaths/min	Y	2	11	2019	10:00	2	11	2019	10:15
42109	99990001	RESPMN15	16	breaths/min	Y	2	11	2019	10:00	2	11	2019	10:15
42109	99990001	HRMAX30	77	beats/min	Y	2	11	2019	10:00	2	11	2019	10:30
42109	99990001	HRMIN30	73	beats/min	Y	2	11	2019	10:00	2	11	2019	10:30
42109	99990001	HRMN30	71	beats/min	Y	2	11	2019	10:00	2	11	2019	10:30
42109	99990001	RESPMAX30	18	breaths/min	Y	2	11	2019	10:00	2	11	2019	10:30
42109	99990001	RESPMIN30	13	breaths/min	Y	2	11	2019	10:00	2	11	2019	10:30
42109	99990001	RESPMN30	15	breaths/min	Y	2	11	2019	10:00	2	11	2019	10:30
42109	99990001	STEPCO	5180	NONE	Y	2	11	2019		2	11	2019	
42109	99990001	ACT_WALK	75	min	Y	2	11	2019		2	11	2019	
42109	99990001	ACT_STAND	124	min	Y	2	11	2019		2	11	2019	
42109	99990001	ACT_SIT	360	min	Y	2	11	2019		2	11	2019	
42109	99990001	ACT_LYNG	780	min	Y	2	11	2019		2	11	2019	
42109	99990001	DIST_DUR	14	min	Y	2	11	2019	13:01	2	11	2019	13:15
42109	99990001	DIST_COV	1180	m	Y	2	11	2019	13:01	2	11	2019	13:15
42109	99990001	DIST_STP	1333	NONE	Y	2	11	2019		2	11		

Example only

PHUSE EU Connect 2022

„Softlanding“ New Data Sources - Our Approach

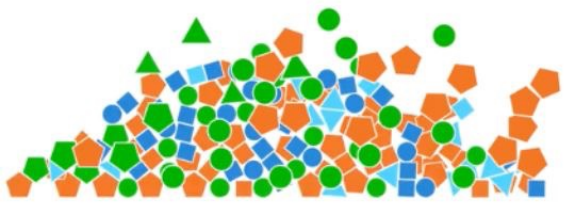
Advanced Ana**ly**tics Platform for the **C**linical Data **E**nvironment (ALYCE)

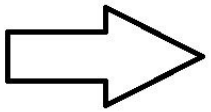
Fulfill patient data protection regulations and adapt to regulated environment

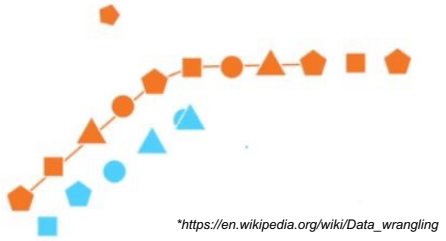
Provide elastic storage, retrieval and processing of large volume, velocity and variety of data

Offer innovative analytics tools to increase analysis speed

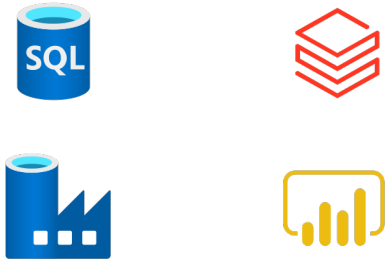
Thereby generate insights faster and more efficiently, increase operational efficiency and deliver new business value



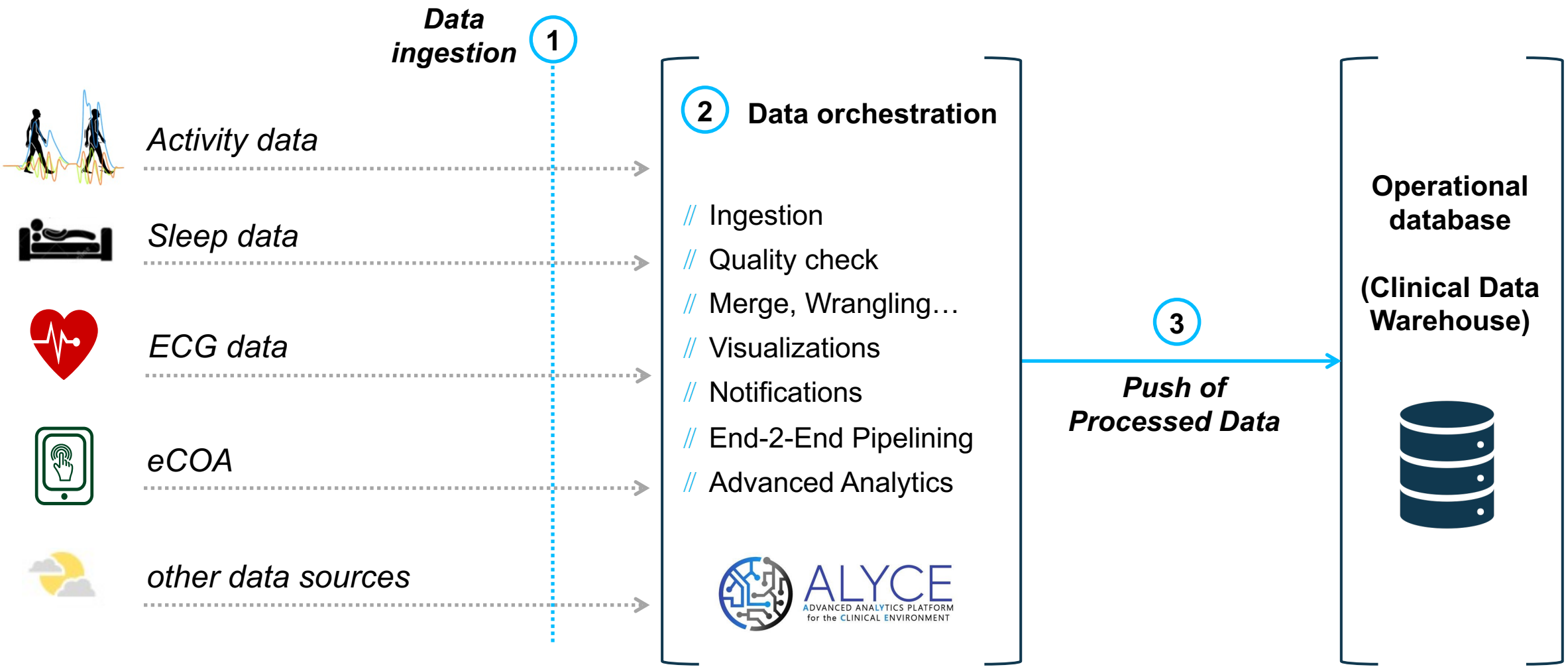




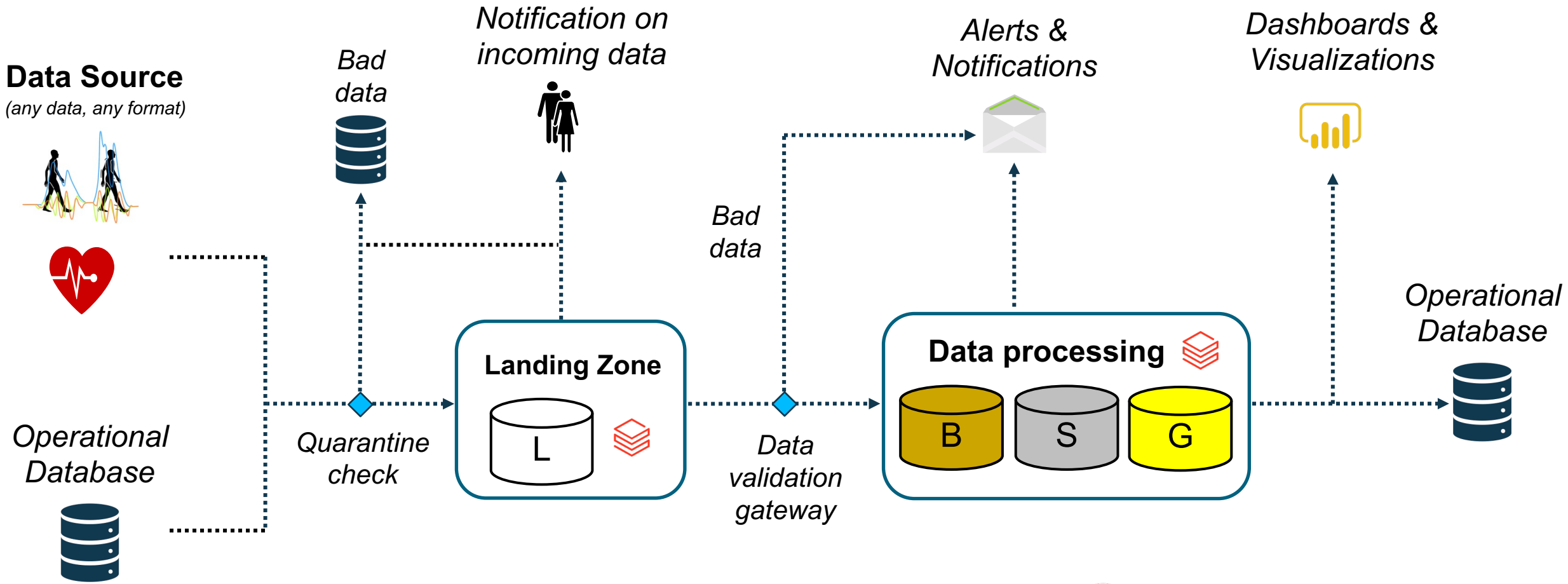
*https://en.wikipedia.org/wiki/Data_wrangling



Ingesting / Wrangling / Analysing new data sources in native format through modern cloud technologies and services

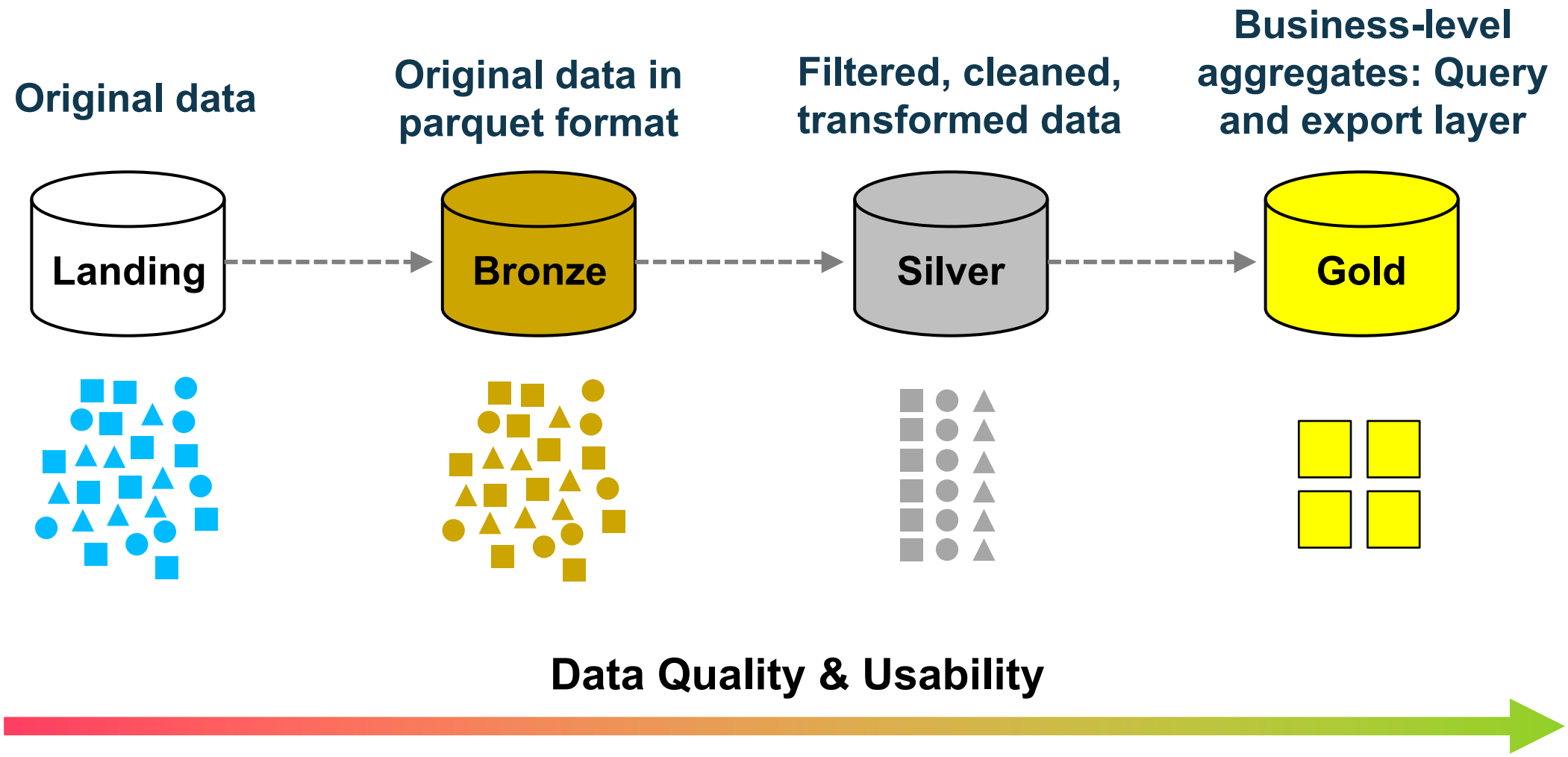


Dataflow – ELTL Approach (Extract, Load, Transform, Load)



End-to-End Pipeline Automation

Staging Approach in the End-2-End Dataflow



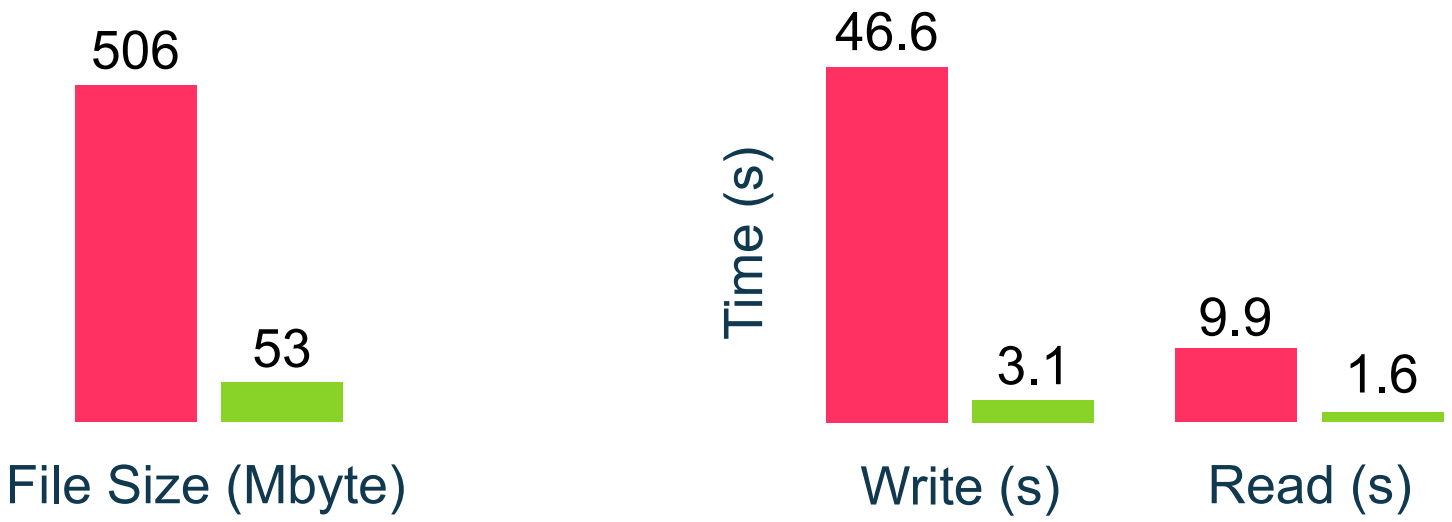
Example – File Size – CSV versus Parquet

// Parquet organizes the data in columns (Columnar Storage) - Binary, efficient compression

// CSV organizes the data row-based (Row Storage)

// Parquet maximize effectiveness of querying the data

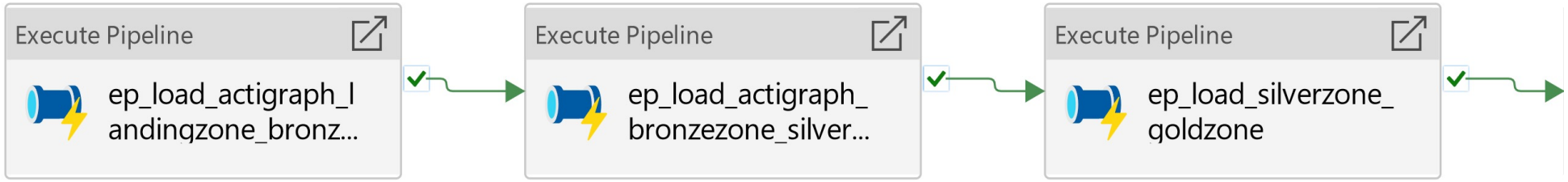
We tested performance of CSV versus Parquet to store data with ECG sample data



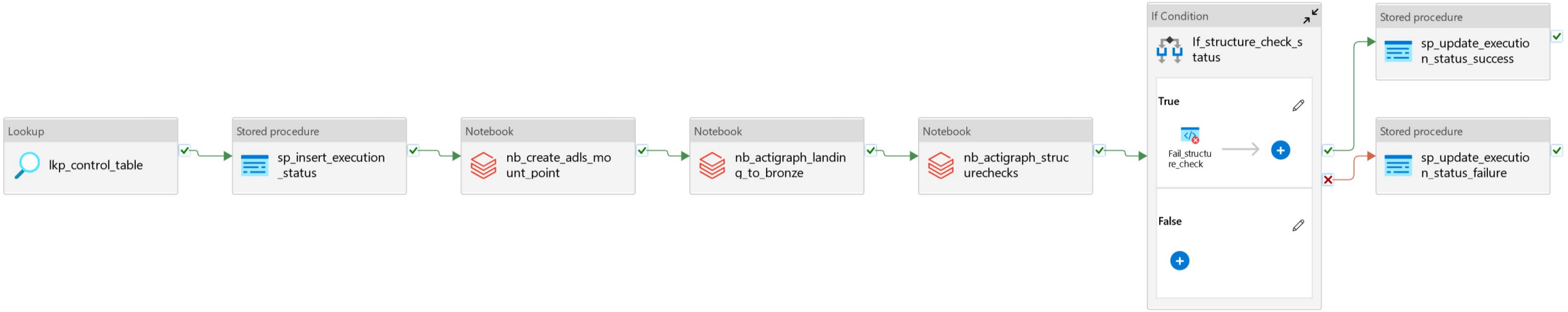
Note that the compression ratio depends on the content of the data, and I/O timings might vary at each attempt.

ELTL Orchestration and Pipelining using Azure Data Factory

ELTL Orchestration e.g. scheduling and connecting of various pipelines



Pipelining of different activities e.g. Ingestion, Check, Wrangling, Movement of Data



Azure Databricks & PySpark for ELTL

```
dbutils.widgets.text("databricks_metadata_path", "")dbutils.widgets.text("databricks_metadata_mnt",
"")dbutils.widgets.text("databricks_landing_path", "")dbutils.widgets.text("databricks_landing_mnt",
"")dbutils.widgets.text("databricks_bronze_mnt", "")dbutils.widgets.text("databricks_bronze_path",
"")dbutils.widgets.text("databricks_actigraph_silver_path", "")dbutils.widgets.text("databricks_silver_mnt",
"")dbutils.widgets.text("databricks_gold_mnt", "")dbutils.widgets.text("databricks_gold_path", "")stalyce_brmnt =
dbutils.widgets.get("databricks_bronze_mnt")stalyce_brpth =
dbutils.widgets.get("databricks_bronze_path")stalyce_slagpth =
dbutils.widgets.get("databricks_actigraph_silver_path")stalyce_slmnt =
dbutils.widgets.get("databricks_silver_mnt")stalyce_mdmnt =
dbutils.widgets.get("databricks_metadata_mnt")stalyce_mdpth =
dbutils.widgets.get("databricks_metadata_path")stalyce_ldmnt =
dbutils.widgets.get("databricks_landing_path")stalyce_ldpth =
dbutils.widgets.get("databricks_landing_mnt")stalyce_glmnt = dbutils.widgets.get("databricks_gold_mnt")stalyce_glpth
= dbutils.widgets.get("databricks_gold_path")
for file in received_files: in_path = BRONZE_PATH + file out_path = 'dbfs:' + SILVER_PATH + file print('Silver path for ' +
file + ': ' + out_path) df = spark.read.format("delta").load(in_path) # only load latest data distinct_ingestdates =
df.select('p_ingestdate').distinct().agg(max_(col("p_ingestdate"))) df =
df.filter(col('p_ingestdate')==distinct_ingestdates.collect()[0][0]) # drop headers df = drop_header_rows(df, metadata)
# convert data types col_count_before = len(df.columns) print('Original number of columns: ' + str(col_count_before))
df = convert_data_types_based_on_metadata(df, metadata) df = convert_timestamp_data_types(df, metadata)
col_count_conversion = len(df.columns) print('Number of columns after data conversion: ' + str(col_count_conversion))
# map the study number df = df.drop("study_number") df = df.withColumnRenamed('STUDYID', 'study_number') # map
the subject_id to actual subject ID df = df.withColumnRenamed('SUBJECT', 'subject_id') # drop the duplicated values
where all columns are the same count_before = df.count() df = df.dropDuplicates([c for c in df.columns if c not in
{'filepath'}]) count_after = df.count() print('Original number of rows: ' + str(count_before)) print('After deduplication: ' +
str(count_after)) print('Deleted duplicate rows: ' + str(count_before-count_after)) df = df.drop("value", "ingesttime",
"filename", 'p_ingestdate') # these should always be dropped col_count_drop = len(df.columns) print('Number of
columns after dropping: ' + str(col_count_drop)) # add the new ingest time df = df.withColumn('RefinedTime',
current_timestamp()) col_count_after = len(df.columns) print('Number of columns after adding: ' +
str(col_count_after)) # reorder columns df = reorder_columns_based_on_metadata(df, metadata) # write to delta
df.write.format("delta").partitionBy('subject_id').mode("overwrite").save(out_path)
```

Example only

- // Python API for Apache Spark
- // Distributed computing framework
- // Optimized for high-volume and high-velocity data processing
- // In-built parallel processing and optimization
- // Spark structured streaming to process incoming data

Monitoring incoming device data using interactive dashboards

- Enable Clinical Trial Teams to monitor incoming data
- Send notifications / alerts if a certain threshold in the data has been reached
- Follow the patient's journey more frequently

PHUSE EU Connect 2022

Summary

Summary

// DHT opens up possibilities for improved trial designs

// Large volume, velocity and variety of (new) data

Using Cloud Computing Platforms can enable:

// Handling new data sources with rich and complex datasets

// Ingesting data more frequently and efficiently from different storage locations

// Processing the data with modern languages optimized for rich and complex data

// Benefit from state-of-the-art interactive visualization-technology

// Automizing the End-2-End dataflow by using pipelining-technology

ALYCE
ADVANCED ANALYTICS PLATFORM for the CLINICAL ENVIRONMENT

Thank you!

Looking forward to CONNECT.

Stephan.Cichos@Bayer.com

or try

LinkedIn - Stephan Cichos