

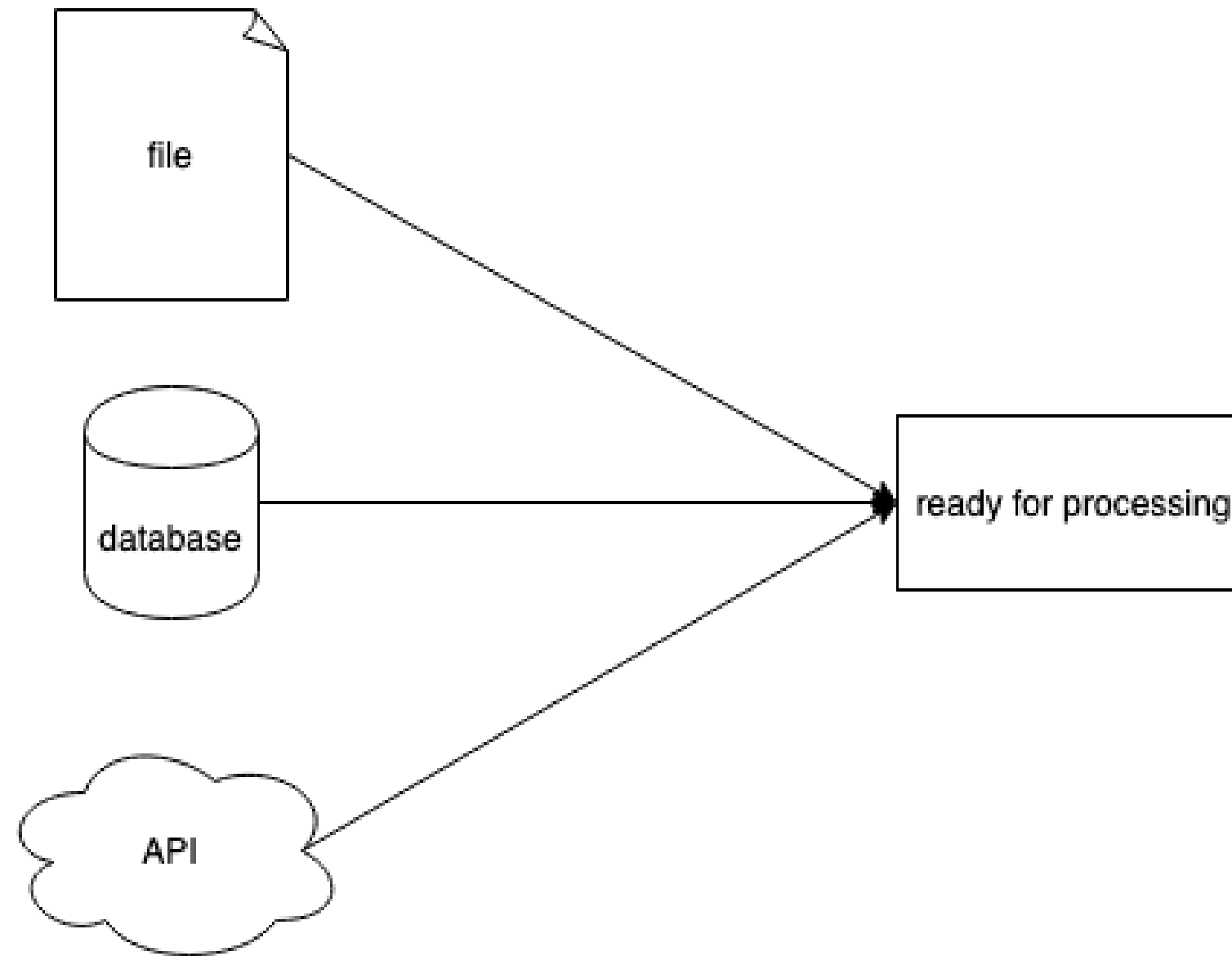
Extract

INTRODUCTION TO DATA ENGINEERING



Vincent Vankrunkelsven
Data Engineer @ DataCamp

Extracting data: what does it mean?



Extract from text files

Unstructured

- Plain text
- E.g. chapter from a book

```
Call me Ishmael. Some years ago—never  
mind how long precisely—having little or  
no money in my purse, and nothing particular  
to interest me on shore, I thought ....
```

Flat files

- Row = record
- Column = attribute
- E.g. `.tsv` or `.csv`

```
Year,Make,Model,Price  
1997,Ford,E350,3000.00  
1999,Chevy,"Venture Extended Edition",4900.00  
1999,Chevy,"Venture Extended Edition",5000.00  
1996,Jeep,Grand Cherokee,4799.00
```

JSON

- JavaScript Object Notation
- Semi-structured
- Atomic
 - number
 - string
 - boolean
 - null
- Composite
 - array
 - object

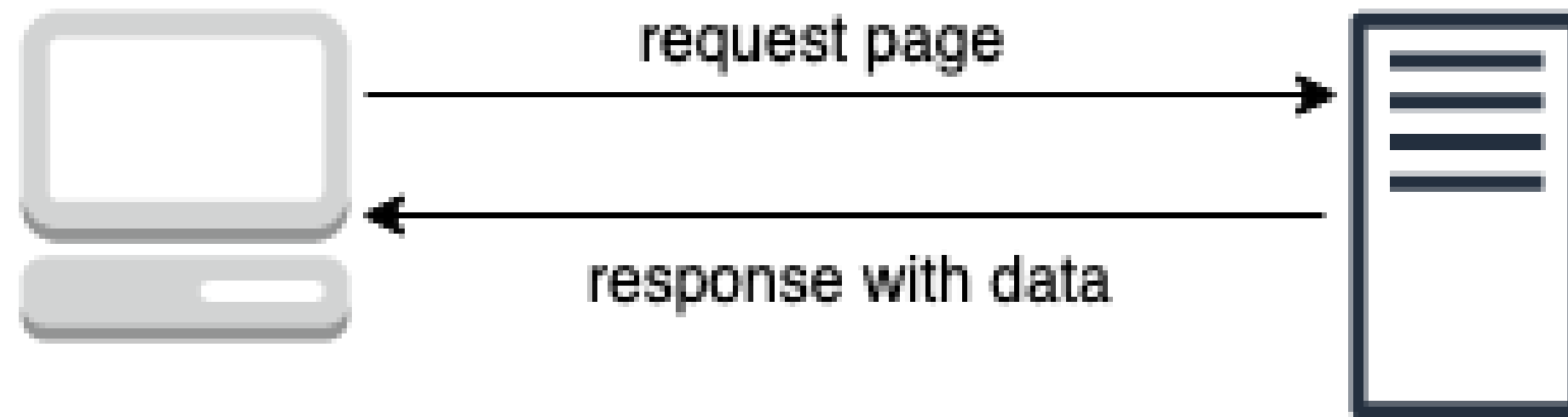
```
{  
  "an_object": {  
    "nested": [  
      "one",  
      "two",  
      "three",  
      {  
        "key": "four"  
      }  
    ]  
  }  
}
```

```
import json  
result = json.loads('{"key_1": "value_1",  
                    "key_2": "value_2"}')  
  
print(result["key_1"])
```

```
value_1
```

Data on the Web

Requests



Example

1. *Browse to Google*
2. *Request to Google Server*
3. *Google responds with web page*

Data on the Web through APIs

- Send data in JSON format
- API: application programming interface
- Examples
 - Twitter API

```
{ "statuses": [{ "created_at": "Mon May 06 20:01:29 +0000 2019", "text": "this is a tweet"}] }
```

- Hackernews API

```
import requests

response = requests.get("https://hacker-news.firebaseio.com/v0/item/16222426.json")
print(response.json())
```

```
{'by': 'neis', 'descendants': 0, 'id': 16222426, 'score': 17, 'time': 1516800333, 'title': .... }
```

Data in databases

Applications databases

- Transactions
- Inserts or changes
- OLTP
- Row-oriented

Analytical databases

- OLAP
- Column-oriented

Extraction from databases

Connection string/URI

```
postgresql://[user[:password]@][host][:port]
```

Use in Python

```
import sqlalchemy
connection_uri = "postgresql://repl:password@localhost:5432/pagila"
db_engine = sqlalchemy.create_engine(connection_uri)

import pandas as pd
pd.read_sql("SELECT * FROM customer", db_engine)
```


Let's practice!

INTRODUCTION TO DATA ENGINEERING

Transform

INTRODUCTION TO DATA ENGINEERING



Vincent Vankrunkelsven
Data Engineer @ DataCamp

Kind of transformations

customer_id	email	state	created_at
1	jane.doe@theweb.com	New York	2019-01-01 07:00:00

- Selection of attribute (e.g. 'email')
- Translation of code values (e.g. 'New York' -> 'NY')
- Data validation (e.g. date input in 'created_at')
- Splitting columns into multiple columns
- Joining from multiple sources

An example: split (Pandas)

customer_id	email	username	domain
1	jane.doe@theweb.com	jane.doe	theweb.com

```
customer_df # Pandas DataFrame with customer data

# Split email column into 2 columns on the '@' symbol
split_email = customer_df.email.str.split("@", expand=True)

# At this point, split_email will have 2 columns, a first
# one with everything before @, and a second one with
# everything after @

# Create 2 new columns using the resulting DataFrame.
customer_df = customer_df.assign(
    username=split_email[0],
    domain=split_email[1],
)
```

Transforming in PySpark

Extract data into PySpark

```
import pyspark.sql

spark = pyspark.sql.Session.builder.getOrCreate()

spark.read.jdbc("jdbc:postgresql://localhost:5432/pagila",
                "customer",
                properties={"user": "repl", "password": "password"})
```

An example: join

A new ratings table

customer_id	film_id	rating
1	2	1
2	1	5
2	2	3
...

The customer table

customer_id	first_name	last_name	...
1	Jane	Doe	...
2	Joe	Doe	...
...

customer_id overlaps with ratings table

An example: join (PySpark)

```
customer_df # PySpark DataFrame with customer data
ratings_df # PySpark DataFrame with ratings data

# Groupby ratings
ratings_per_customer = ratings_df.groupBy("customer_id").mean("rating")

# Join on customer ID
customer_df.join(
    ratings_per_customer,
    customer_df.customer_id==ratings_per_customer.customer_id
)
```

Let's practice!

INTRODUCTION TO DATA ENGINEERING

Loading

INTRODUCTION TO DATA ENGINEERING



Vincent Vankrunkelsven
Data Engineer @ DataCamp

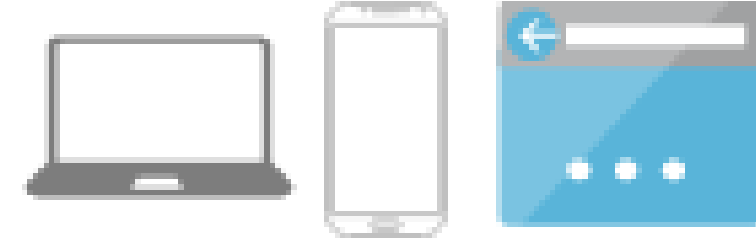
Analytics or applications databases

Analytics



- Aggregate queries
- Online analytical processing (OLAP)

Applications



- Lots of transactions
- Online transaction processing (OLTP)

Column- and row-oriented

Analytics

- Column-oriented

name	diameter (cm)	weight (g)
apple	10	100
grape	2	10

- Queries about subset of columns
- Parallelization

Applications

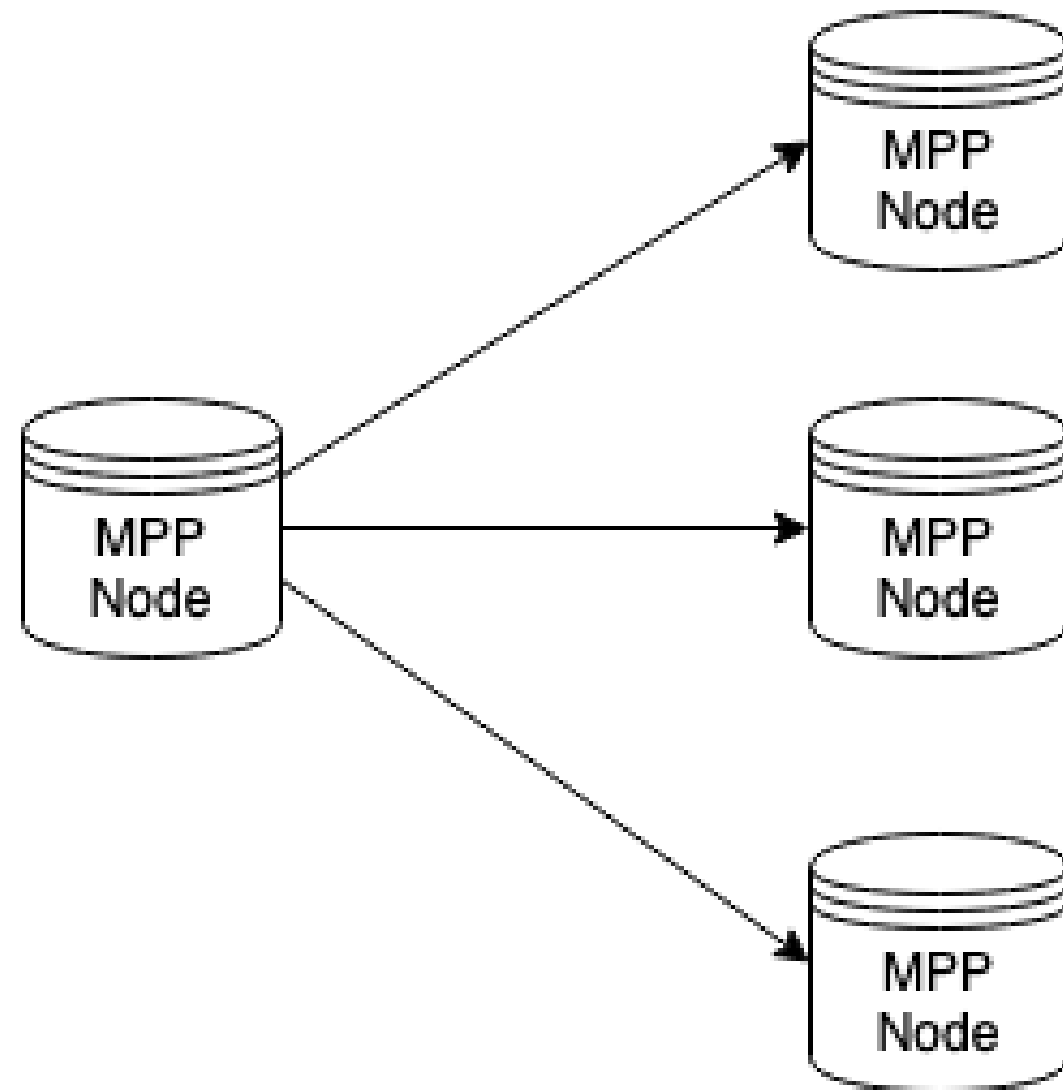
- Row-oriented

name	diameter (cm)	weight (g)
apple	10	100
grape	2	10

- Stored per record
- Added per transaction
- E.g. adding customer is fast

MPP Databases

Massively Parallel Processing Databases



- Amazon Redshift
- Azure SQL Data Warehouse
- Google BigQuery

An example: Redshift

Load from file to columnar storage format

```
# Pandas .to_parquet() method
df.to_parquet("./s3://path/to/bucket/customer.parquet")

# PySpark .write.parquet() method
df.write.parquet("./s3://path/to/bucket/customer.parquet")
```

```
COPY customer
FROM 's3://path/to/bucket/customer.parquet'
FORMAT as parquet
...
```

Load to PostgreSQL

```
pandas.to_sql()
```

```
# Transformation on data
recommendations = transform_find_recommendations(ratings_df)

# Load into PostgreSQL database
recommendations.to_sql("recommendations",
                        db_engine,
                        schema="store",
                        if_exists="replace")
```

Let's practice!

INTRODUCTION TO DATA ENGINEERING

Putting it all together

INTRODUCTION TO DATA ENGINEERING



Vincent Vankrunkelsven
Data Engineer @ DataCamp

The ETL function

```
def extract_table_to_df(tablename, db_engine):
    return pd.read_sql("SELECT * FROM {}".format(tablename), db_engine)

def split_columns_transform(df, column, pat, suffixes):
    # Converts column into str and splits it on pat...

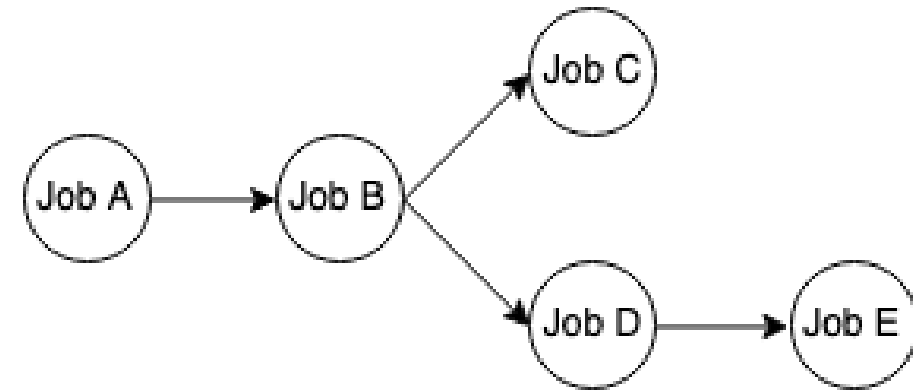
def load_df_into_dwh(film_df, tablename, schema, db_engine):
    return pd.to_sql(tablename, db_engine, schema=schema, if_exists="replace")

db_engines = { ... } # Needs to be configured
def etl():
    # Extract
    film_df = extract_table_to_df("film", db_engines["store"])
    # Transform
    film_df = split_columns_transform(film_df, "rental_rate", ".", ["_dollar", "_cents"])
    # Load
    load_df_into_dwh(film_df, "film", "store", db_engines["dwh"])
```

Airflow refresher



- Workflow scheduler
- Python
- DAGs



- Tasks defined in operators (e.g. `BashOperator`)

Scheduling with DAGs in Airflow

```
from airflow.models import DAG
```

```
dag = DAG(dag_id="sample",  
          ...,  
          schedule_interval="0 0 * * *")
```

```
# cron  
# .----- minute (0 - 59)  
# | .----- hour (0 - 23)  
# | | .----- day of the month (1 - 31)  
# | | | .----- month (1 - 12)  
# | | | | .----- day of the week (0 - 6)  
# * * * * * <command>
```

```
# Example
```

```
0 * * * * # Every hour at the 0th minute
```

cf. <https://crontab.guru>

The DAG definition file

```
from airflow.models import DAG
from airflow.operators.python_operator import PythonOperator

dag = DAG(dag_id="etl_pipeline",
          schedule_interval="0 0 * * *")

etl_task = PythonOperator(task_id="etl_task",
                          python_callable=etl,
                          dag=dag)

etl_task.set_upstream(wait_for_this_task)
```

The DAG definition file


```
from airflow.models import DAG
from airflow.operators.python_operator import PythonOperator

...

etl_task.set_upstream(wait_for_this_task)
```

Saved as `etl_dag.py` in `~/airflow/dags/`

Airflow UI

 Airflow

DAGs

Data Profiling ▾

Browse ▾

Admin ▾





























Docs ▾

About ▾

2019-08-05 17:53:00 UTC

DAGs

Search:

		DAG	Schedule	Owner	Recent Tasks 	Last Run 	DAG Runs 	Links
	 Off	etl_pipeline	00 ***		         		  	        

Showing 1 to 1 of 1 entries

«

<

1

>

»

Hide Paused DAGs

Let's practice!

INTRODUCTION TO DATA ENGINEERING