

# Popularne formaty plików w codzienności inżyniera danych

Natalia Warszewska



20 grudzień 2023, Bydgoszcz



# Natalia Warszewska

✉ [natalia.warszewska@datacommunity.pl](mailto:natalia.warszewska@datacommunity.pl)

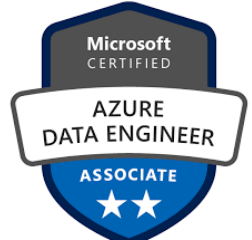
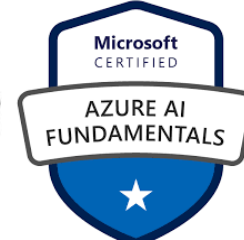
in [/natwarszewska](#)



Senior Software Engineer



Member of SQLDay Council





**Data  
Community**

MessagePack

HL7

EDI

Arrow Flight

**Avro**

RDF

Feather

SAS

**JSON**

**CSV**

ZIP

HDF5

**XML**

XLSX

**Parquet**

ARFF

PDF

TXT

JSONL

Binary

**ORC**

XLS

SQL

**YAML**

DICOM

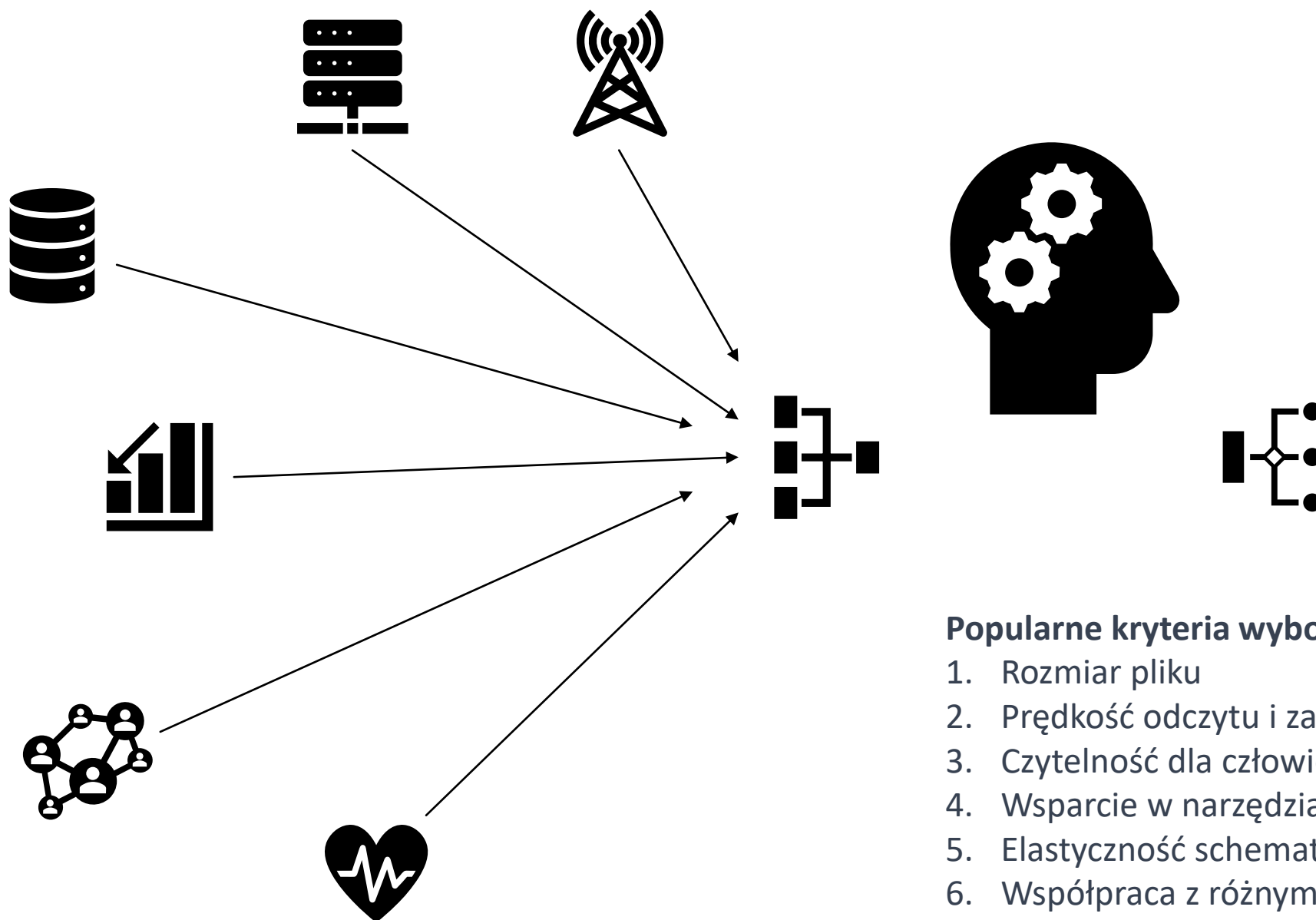
SQLite

Protobuf

GeoJSON

***Format pliku** to sposób organizacji danych w pliku, który określa strukturę, sposób przechowywania informacji oraz reguły odczytu i zapisu danych. Jest to zbiór ustalonych standardów, konwencji i specyfikacji, które określają, jak dane są reprezentowane i przechowywane w danym pliku.*

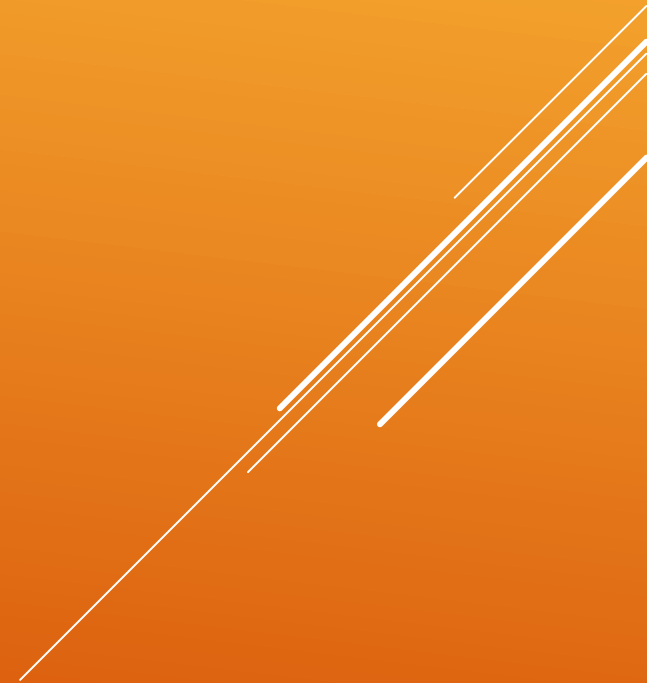
*Format pliku może obejmować informacje dotyczące nagłówków, metadanych, a także sposobu kodowania i reprezentacji różnych typów danych.*



#### Popularne kryteria wyboru:

1. Rozmiar pliku
2. Prędkość odczytu i zapisu
3. Czytelność dla człowieka
4. Wsparcie w narzędziach
5. Elastyczność schematu
6. Współpraca z różnymi językami programowania

# Popularne formaty



## CSV (Comma-Separated Values)

CSV to format pliku tekstowego używanego do przechowywania danych w formie tabelarycznej, gdzie każda linia reprezentuje rekord, a pola w rekordzie są oddzielone przecinkami (stąd nazwa "Comma-Separated Values").

### Zalety:


- prostota i uniwersalność,
- czytelność,
- wsparcie w wielu narzędziach

### Wady:

- brak standardu (różne separatory w praktyce),
- mniej efektywne dla dużych zbiorów danych
- znaki specjalne
- problem przy danych hierarchicznych

### CSV (Comma-Separated Values):

CSV

 Copy code

```
Name, Age, City  
John, 28, New York  
Alice, 24, San Francisco  
Bob, 32, Los Angeles
```

## JSON (JavaScript Object Notation)

JSON to format otwarty, lekki i niezależny od języka, służący do wymiany danych. Jest to zapis danych w formie tekstowej, oparty na konwencji obiektowej znanego z języka JavaScript. Struktura danych w formacie JSON opiera się na parach klucz-wartość oraz zagnieżdżonych obiektach i tablicach.

**Struktura JSON:** pary klucz-wartość, tablice.

### Zalety:


- elastyczność,
- uniwersalność  
(podobny ale niezależny od JavaScript),
- czytelność dla człowieka,
- Popularność w API

### Wady:

- większy rozmiar pliku w porównaniu do binarnych formatów,
- brak standardu daty

### JSON (JavaScript Object Notation):

json

 Copy code

```
{
  "people": [
    { "name": "John", "age": 28, "city": "New York" },
    { "name": "Alice", "age": 24, "city": "San Francisco" },
    { "name": "Bob", "age": 32, "city": "Los Angeles" }
  ]
}
```



## Avro

Avro to otwarty, binarny format danych opracowany w ramach projektu Apache. Jest to kompaktowy i efektywny sposób reprezentacji danych, często używany w systemach przetwarzania danych rozproszonych, takich jak Apache Hadoop. Avro obsługuje ewolucję schematu, co oznacza, że dane mogą ewoluować bez konieczności zatrzymywania systemu.

### Zalety:

- kompresja,
- efektywność,
- wsparcie dla ewolucji schematu,
- wsparcie w wielu językach,
- Integracja z Apache Hadoop (Kafka, Spark)


### Wady:

- Mniej czytelny dla człowieka

### Avro:

- Format binarny, a poniżej jest przykład Avro w formie JSON dla zobrazowania struktury:

json

 Copy code

```
{  
  "type": "record", "name": "Person", "fields": [  
    {  
      "name": "name", "type": "string",  
    },  
    {  
      "name": "age", "type": "int",  
    },  
    {  
      "name": "city", "type": "string",  
    }  
  ]  
}
```

## Parquet

Parquet to format przechowywania danych w formie kolumnowej, który został stworzony w ramach projektu Apache. Jest to format binarny, skompresowany, zaprojektowany z myślą o efektywnym przechowywaniu dużych zbiorów danych, szczególnie w kontekście przetwarzania danych rozproszonych.

### Zalety:

- efektywność przetwarzania kolumnowego,
- wsparcie dla dużej ilości danych
- wsparcie dla ewolucji schematu (umożliwia zmiany w strukturze danych bez utraty zgodności w systemie)
- Kompresja, optymalizacja przetwarzania,
- Współpraca z narzędziami analitycznymi
- Wieloplatformowość


### Wady:

- Mniej czytelny dla człowieka

### Parquet:

- Ponieważ Parquet to format binarny, reprezentacja w formie tekstu jest trudna. Poniżej znajduje się przykład schematu w języku Apache Arrow, który jest również stosowany w Parquet:

json

 Copy code

```
[  
  {"name": "name", "type": "string"},  
  {"name": "age", "type": "int"},  
  {"name": "city", "type": "string"}  
]
```

## ORC (Optimized Row Columnar)

ORC to format przechowywania danych w formie kolumnowej, opracowany w ramach projektu Apache Hive (część ekosystemu Apache Hadoop). Jest zoptymalizowany pod kątem efektywnego przechowywania i przetwarzania dużych zbiorów danych w systemach rozproszonych.

### Zalety:

- efektywność,
- niska latencja,
- Wsparcie dla ewolucji schematu,
- Popularność w hurtowniach danych i systemach rozproszonych (Apache Hive, Apache Spark)


### Wady:

- Mniej czytelny dla człowieka

### ORC (Optimized Row Columnar):

- Format binarny, więc poniżej jest przykład schematu w formie tekstu:

json

 Copy code

```
{
  "type": "struct",
  "fields": [
    {"name": "name", "type": "string"},
    {"name": "age", "type": "int"},
    {"name": "city", "type": "string"}
  ]
}
```

## XML (eXtensible Markup Language)

XML to uniwersalny język znaczników stosowany do reprezentacji strukturalnych danych w formie tekstowej. Jest to format otwarty, oparty na hierarchicznym modelu drzewa, w którym dane są opisane za pomocą znaczników, a struktura hierarchiczna pozwala reprezentować różne rodzaje danych.

Struktura oparta na tagach.

### Zalety:

- czytelność,
- wsparcie dla hierarchicznych danych.

### Wady:

- większy rozmiar pliku w porównaniu do niektórych formatów.

### XML (eXtensible Markup Language):


```
xml Copy code  
  
<people>  
  <person>  
    <name>John</name>  
    <age>28</age>  
    <city>New York</city>  
  </person>  
  <person>  
    <name>Alice</name>  
    <age>24</age>  
    <city>San Francisco</city>  
  </person>  
  <person>  
    <name>Bob</name>  
    <age>32</age>  
    <city>Los Angeles</city>  
  </person>  
</people>
```

## YAML (YAML Ain't Markup Language):

YAML to format notacji dla danych, często używany do konfiguracji plików i wymiany danych pomiędzy programami. Jest to format tekstu, który zapisuje dane w sposób czytelny dla ludzi, co sprawia, że jest popularny w konfiguracjach, skryptach i plikach konfiguracyjnych.


### YAML (YAML Ain't Markup Language):

yaml

 Copy code

```
people:
  - name: John
    age: 28
    city: New York
  - name: Alice
    age: 24
    city: San Francisco
  - name: Bob
    age: 32
    city: Los Angeles
```

<https://swagger.io/specification/v2/>

 **Swagger Editor**  
Supported by SMARTBEAR

File ▾ Edit ▾ Insert ▾ Generate Server ▾ Generate Client ▾ About ▾

Try our new Editor ↗

```
1 openapi: 3.0.3
2 info:
3   title: Swagger Petstore - OpenAPI 3.0
4   description: |-
5     This is a sample Pet Store Server based on the OpenAPI 3.0 specification. You
6     can find out more about
7     Swagger at [https://swagger.io](https://swagger.io). In the third iteration of
8     the pet store, we've switched to the design first approach!
9     You can now help us improve the API whether it's by making changes to the
10    definition itself or to the code.
11    That way, with time, we can improve the API in general, and expose some of the
12    new features in OAS3.
13
14    _If you're looking for the Swagger 2.0/OAS 2.0 version of Petstore, then click
15    [here](https://editor.swagger.io/?url=https://petstore.swagger.io/v2/swagger
16    .yaml). Alternatively, you can load via the `Edit > Load Petstore OAS 2.0`
17    menu option!_
18
19    Some useful links:
20    - [The Pet Store repository](https://github.com/swagger-api/swagger-petstore)
21    - [The source API definition for the Pet Store](https://github.com/swagger-api
22    /swagger-petstore/blob/master/src/main/resources/openapi.yaml)
23
24 termsOfService: http://swagger.io/terms/
25 contact:
26   email: apiteam@swagger.io
27 license:
28   name: Apache 2.0
29   url: http://www.apache.org/licenses/LICENSE-2.0.html
30 version: 1.0.11
31 externalDocs:
32   description: Find out more about Swagger
```

## Swagger Petstore - OpenAPI 3.0 1.0.11

### OAS 3.0

This is a sample Pet Store Server based on the OpenAPI 3.0 specification. You can find out more about Swagger at <https://swagger.io>. In the third iteration of the pet store, we've switched to the design first approach! You can now help us improve the API whether it's by making changes to the definition itself or to the code. That way, with time, we can improve the API in general, and expose some of the new features in OAS3.

*If you're looking for the Swagger 2.0/OAS 2.0 version of Petstore, then click [here](#). Alternatively, you can load via the **Edit > Load Petstore OAS 2.0** menu option!*

Some useful links:

- [The Pet Store repository](https://github.com/swagger-api/swagger-petstore)
- [The source API definition for the Pet Store](https://github.com/swagger-api/swagger-petstore/blob/master/src/main/resources/openapi.yaml)

[Terms of service](#)

[Contact the developer](#)


[Apache 2.0](#)


# Różnice w strukturze i scenariusze zastosowania




## Różnice w Strukturze (wpływ na czytelność i wydajność)


### **płaskie** np. CSV

 Prosta struktura płaska sprawia, że CSV jest czytelny dla człowieka, zwłaszcza dla prostych danych tablicowych.


 Prosta struktura CSV sprawia, że jest efektywny pod względem czasu odczytu/zapisu dla prostych danych tablicowych.


### **hierarchiczne** np. JSON, XML

 Hierarchiczna struktura JSON i XML jest bardziej czytelna dla ludzi w przypadku danych zagnieżdżonych i bardziej złożonych struktur.

 W przypadku bardziej złożonych danych, JSON i XML mogą być bardziej zasobożerne pod względem przetwarzania.

### **kolumnowe** np. Parquet, ORC

 Struktura kolumnowa jest skierowana bardziej do efektywności przetwarzania niż czytelności dla człowieka.

 Struktura kolumnowa w Parquet i ORC może przyspieszyć operacje analityczne i zapytania, zwłaszcza gdy wymagane są tylko pewne kolumny.



## Wybór Formatu zależnie od Sytuacji

### Prosta Analiza Tabelaryczna:

- CSV może być najbardziej odpowiedni do prostych analiz tabelarycznych.

### Dane Zagnieżdżone i Strukturalne:

- JSON, XML lub YAML są bardziej odpowiednie, gdy dane są zagnieżdżone lub posiadają bardziej skomplikowaną strukturę.

### Przetwarzanie Dużych Zbiorów Danych Analitycznych:

- Parquet, Avro i ORC są optymalne do analizy dużych zbiorów danych analitycznych.

### Przetwarzanie Strumieniowe w Systemach Big Data:

- Parquet, Avro, ORC, JSON lub XML w zależności od specyfiki systemu i wymagań przetwarzania strumieniowego.

### Konfiguracja Aplikacji:

- YAML jest często stosowany w konfiguracji aplikacji ze względu na swoją czytelność dla ludzi.

**DEMO**





MessagePack

HL7

Avro

EDI

Arrow Flight

Feather

RDF

SAS

JSON

CSV

ZIP

HDF5

XML

XLSX

ARFF

PDF

TXT

Parquet

JSONL

Binary

ORC

XLS

SQL

YAML

DICOM

SQLite

Protobuf

GeoJSON



Q&A