Prof. Dr. Florian Heinz
florian.heinz@sysv.de

Modern Database Concepts

Chapter 2: Semi-Structured Data

OSTBAYERISCHE
TECHNISCHE HOCHSCHULE
REGENSBURG

Classifications of Data

Structured Data

product_id	description	price
17	chocolate bar	0.89
29	dishwasher tabs	3.99

Semi-Structured Data

```
{ "product_id": 17,
  "description" : "chocolate bar",
  "price": 0.89 }
```

Unstructured Data

Structured data is easy to query as it follows a fixed data structure. The meta data (here: the column names) describes the data. In semi-structured data, the metadata is part of the data itself. So, the JSON document shown in the example describes itself. It is easy for machines to search and extract information from this document, but applications have to cope with a flexible schema. Other documents can have other attributes, and in other documents, other data types can be used. In unstructured data, there is no structure at all. What we have is a text from which we first have to extract the information. This is not as easy and safe as for the other two classes of data.

Data Formats for Semi-Structured Data

- CSV - Comma-separated Values

```
17,chocolate bar,0.89
```

- YAML - YAML Ain’t Markup Language
- XML - Extensible Markup Language

```
<product id="17">chocolate bar<price>0.89</price></product>
```

- JSON - JavaScript Object Notation
- BSON - Binary JSON

The formats shows here are often used as data-exchange formats (e.g. via files), for configuration files, or within document databases (e.g., MongoDB uses JSON and BSON)

CSV

Comma-Separated Values

```
product_id,description,price
17,chocolate bar,0.89
29,"dishwasher tabs, 100 pack",3.99
88,"cat food ""Catcat"""
```

Properties of the example CSV above:

Column Separator	,
Column Delimiter	"
Header row	yes
Encoding	UTF-8

When the column separator or the column delimiter is required within a column value, this value is written between column-delimiter characters. When the column delimiter (here: ") is needed, it is doubled.

YAML

YAML Ain't Markup Language (human-readable data-serialization language)

3 data types, nesting via indentation

- scalar: numeric (integer, float), boolean, string, null
- list
- dictionary

```yaml
---
product_id: 17
description: chocolate bar
price: 0.89 # very cheap!
categories:
  - food
  - sweets
manufacturer:
  company: Monsterfood
  country: USA
available: true
size: null
```

Try out at http://www.yamllint.com/

YAML

YAML Advanced Features:

Aliases:

```yaml
---
name: &var1 "Modern Database Concepts"
keywords:
  - Databases
  - *var1
```

Complex keys:

```yaml
---
? [ "Key", "is", "a", "List" ]
: "Value is a string"
```

YAML also has several complex features, e.g. Aliases for referencing other parts of the document or complex (non-scalar) keys.

YAML

YAML Ain't Markup Language (human-readable data-serialization language)

Goals:

- YAML should be easily readable by humans
- YAML data should be portable between programming languages
- YAML should match the native data structures of dynamic languages
- YAML should have a consistent model to support generic tools
- YAML should support one-pass processing
- YAML should be expressive and extensible
- YAML should be easy to implement and use

YAML

YAML Ain't Markup Language (human-readable data-serialization language)

Useful Links:

- Specification: https://yaml.org/spec/1.2.2/
- JSON2YAML: https://www.json2yaml.com/
- YAMLlint: https://www.yamllint.com/