

Week 6 - Data models and validation

Aljaz Medic



Plan for today

1. Exercise Sheet Questions?
2. Quiz
3. Dremel (Mandatory Reading)
4. Old Exam Questions

QUIZ

Data models describe how data is physically stored on disk.

False. (Data models define the logical structure of data, independent of physical storage.)

JSON and XML are represented as a tree structure in memory.

True.

How is XML's data model different from JSON's?

Node labels in JSON are on the edges whereas in XML they are on the nodes

QUIZ

Records and Maps are synonyms for the same structured type.

False. (Records map from strings -> value, Maps from any atomic value -> value.)

Name all structured data types illustrated in the lecture.

Maps, Records, Lists, Sets

Parsing is the process of building a tree in memory from text.

True.

QUIZ

A document can be valid against a certain schema even if it is not well formed.

False. (A document must be well formed before it can be valid.)

Type validation and annotation are the same process.

False. (Validation checks data against a schema; annotation adds type or metadata information to nodes and usually stores them in native types.)

Validating data usually makes subsequent queries slower.

False. (Validation generally improves performance by ensuring predictable data structures.)

QUIZ

Validation ensures that a document is well-formed.

False. (Well-formedness checks the syntax; validation checks conformance to a schema.)

What is the problem with interpreting time intervals.

Months and days cannot be combined since months have variable length. Reference: ISO-8601

Name the cardinality symbols and their meanings.

?, *, + → optional, zero-or-more, one-or-more

QUIZ

Open object types allow extra fields beyond those defined in the schema.

True.

In JSON schema and JSound, specified fields are required by default.

False. (Specified fields in both are optional by default.)

JSON and XML Schema are open by default.

False. (In XML Schema, by default minOccurs and maxOccurs equals 1, making it closed by default.

JSound is closed by default.

JSON Schema is open by default; we forbid extra keys by setting "additionalProperties" to false.)

QUIZ

JSound can define primary keys within arrays of objects.

True.

When defining JSound schema, we make a field mandatory by adding a "!" suffix, similar as in Typescript.

False. (We have to prefix "!")

The lexical space of a type defines the range of possible values it can take.

False. (The lexical space defines the textual representations of values; the value space defines the actual set of values.)

QUIZ

If a value of an "int" field is provided as a string of digits, JSound marks it invalid.

False. (It is in the correct lexical space; "42" is annotated as 42 in memory.)

List all (4) types of XML Information Items covered in class.

Document, Element, Attribute, Text (Character). (There are many more.)

In XML, attributes can have complex types.

False. (Attributes can only have simple atomic types; elements can have complex types.)

QUIZ

What are DataFrames?

Valid datasets with schema requirements:

open object types forbidden; object and array values must have specific types.

Data frames primarily store flat homogeneous data.

False. (They are homogeneous*, but can also store nested data.)

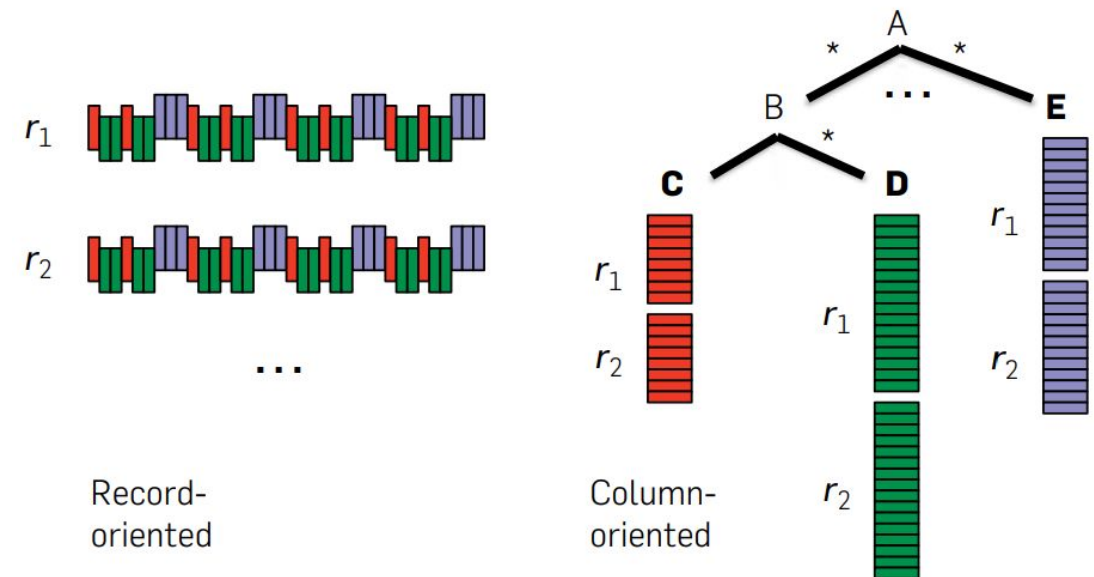
Why are binary formats like Parquet, Dremel and Avro preferred for DataFrames?

They offer immediate improvements in space and performance efficiency.

Dremel

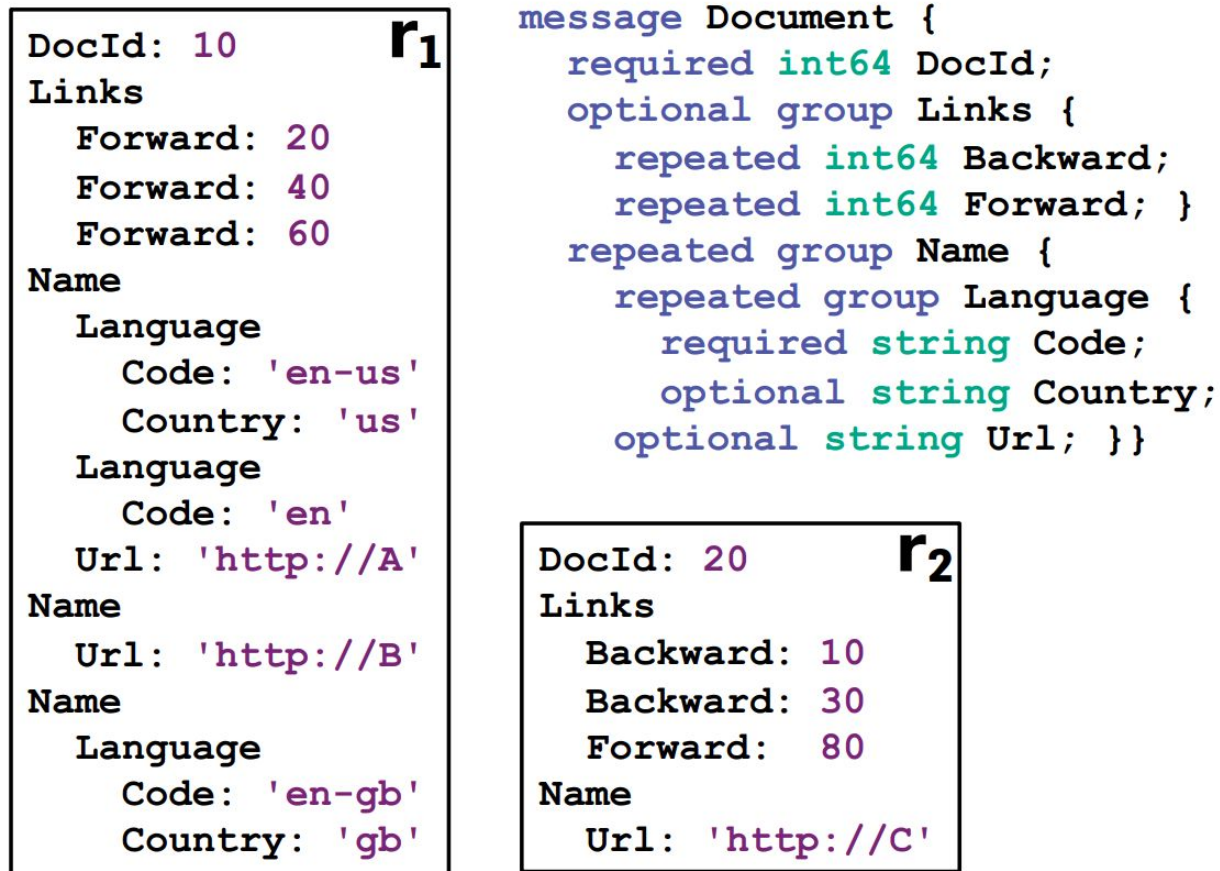
- Developed by Google:
- “Scalable, interactive ad hoc query system for analysis of read-only nested data.”
- Uses columnar storage for nested data
- Goal: store all values of a given field consecutively to improve retrieval efficiency.
- Data model is based on strongly typed nested records, called *Protocol Buffers*
- Has SQL-like querying.

Figure 1. Record-wise vs. columnar representation of nested data.



Dremel - Sample nested records

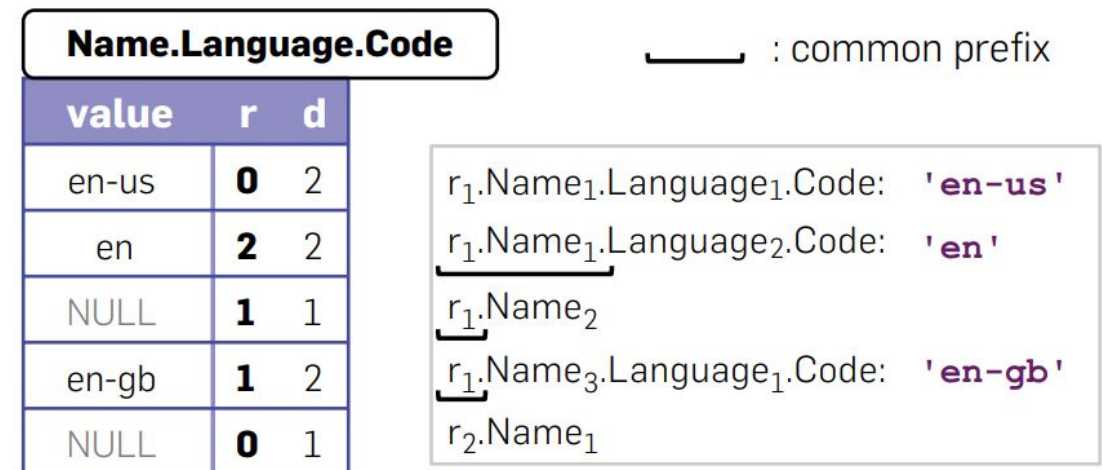
Figure 2. Two sample nested records and their schema.



repetition level: length of common prefix, counting **repeated** fields, including first path component *record_id*.

definition level: total number of **repeated** and **optional** fields, excluding *record_id*.

Figure 4. Repetition and definition levels: delta between paths.



Dremel - Cheatsheet

Figure 2. Two sample nested records and their schema.

```
DocId: 10 r1
Links
  Forward: 20
  Forward: 40
  Forward: 60
Name
  Language
    Code: 'en-us'
    Country: 'us'
  Language
    Code: 'en'
  Url: 'http://A'
Name
  Url: 'http://B'
Name
  Language
    Code: 'en-gb'
    Country: 'gb'
```

```
message Document {
  required int64 DocId;
  optional group Links {
    repeated int64 Backward;
    repeated int64 Forward; }
  repeated group Name {
    repeated group Language {
      required string Code;
      optional string Country; }
    optional string Url; }}
```

```
DocId: 20 r2
Links
  Backward: 10
  Backward: 30
  Forward: 80
Name
  Url: 'http://C'
```

repetition level: length of common prefix, counting **repeated** fields, including first path component *record_id*.

definition level: total number of **repeated** and **optional** fields, excluding *record_id*.

- Common prefix always ends on a repeated field.
- We never count **required** fields.
- If you can write a r,d-pair in the column strip, then you have to write it.
- In a given column stripe, we know the max possible value for **d**. If **d** is set to less than that, it denotes a NULL.
- When you see 0, it's a different record.

Dremel - Sample nested records

Figure 2. Two sample nested records and their schema.

DocId: 10 Links Forward: 20 Forward: 40 Forward: 60 Name Language Code: 'en-us' Country: 'us' Language Code: 'en' Url: 'http://A' Name Url: 'http://B' Name Language Code: 'en-gb' Country: 'gb'	r₁	<pre> message Document { required int64 DocId; optional group Links { repeated int64 Backward; repeated int64 Forward; } repeated group Name { repeated group Language { required string Code; optional string Country; } optional string Url; }} </pre>	DocId: 20 Links Backward: 10 Backward: 30 Forward: 80 Name Url: 'http://C'	r₂
--	----------------------	--	---	----------------------

Figure 3. Column-striped representation of the data in Figure 2.

DocId

value	r	d
10	0	0
20	0	0

Name.Url

value	r	d
http://A	0	2
http://B	1	2
NULL	1	1
http://C	0	2

Links.Forward

value	r	d
20	0	2
40	1	2
60	1	2
80	0	2

Links.Backward

value	r	d
NULL	0	1
10	0	2
30	1	2

Name.Language.Code

value	r	d
en-us	0	2
en	2	2
NULL	1	1
en-gb	1	2
NULL	0	1

Name.Language.Country

value	r	d
us	0	3
NULL	2	2
NULL	1	1
gb	1	3
NULL	0	1

Dremel - Full Reconstruction

Figure 2. Two sample nested records and their schema.

DocId: 10	r₁
Links	
Forward: 20	
Forward: 40	
Forward: 60	
Name	
Language	
Code: 'en-us'	
Country: 'us'	
Language	
Code: 'en'	
Url: 'http://A'	
Name	
Url: 'http://B'	
Name	
Language	
Code: 'en-gb'	
Country: 'gb'	

```

message Document {
  required int64 DocId;
  optional group Links {
    repeated int64 Backward;
    repeated int64 Forward; }
  repeated group Name {
    repeated group Language {
      required string Code;
      optional string Country; }
    optional string Url; }}
  
```

DocId: 20	r₂
Links	
Backward: 10	
Backward: 30	
Forward: 80	
Name	
Url: 'http://C'	

Figure 6. Complete record assembly automaton. Edges are labeled with repetition levels.

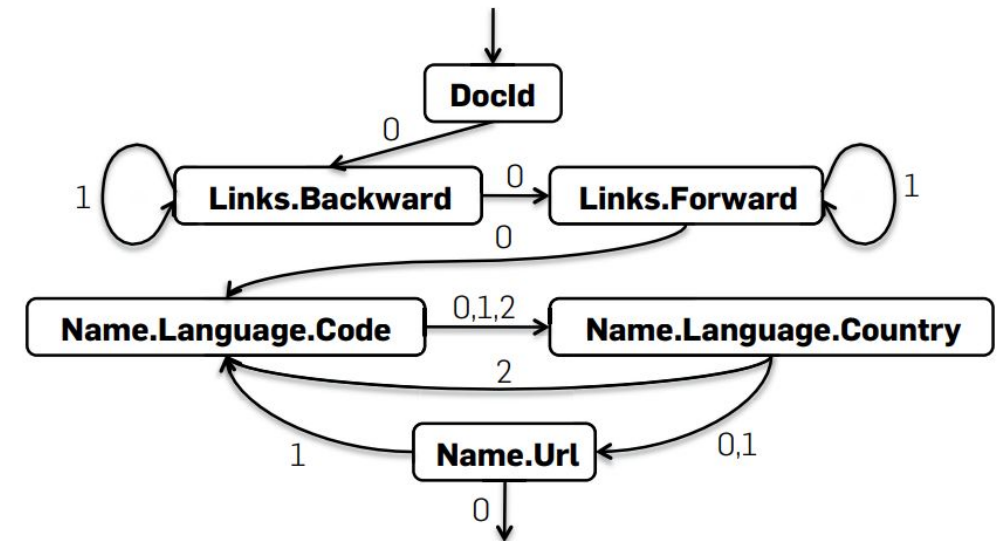


Figure 3. Column-striped representation of the data in Figure 2.

DocId	Name.Url	Links.Forward	Links.Backward
value	value	value	value
r	r	r	r
d	d	d	d
10	http://A	20	NULL
20	http://B	40	10
	NULL	60	30
	http://C	80	

Name.Language.Code	Name.Language.Country
value	value
r	r
d	d
en-us	us
en	NULL
NULL	NULL
en-gb	gb
NULL	NULL

Dremel - Partial Reconstruction

Figure 2. Two sample nested records and their schema.

DocId: 10	r₁
Links	
Forward: 20	
Forward: 40	
Forward: 60	
Name	
Language	
Code: 'en-us'	
Country: 'us'	
Language	
Code: 'en'	
Url: 'http://A'	
Name	
Url: 'http://B'	
Name	
Language	
Code: 'en-gb'	
Country: 'gb'	

```
message Document {
  required int64 DocId;
  optional group Links {
    repeated int64 Backward;
    repeated int64 Forward; }
  repeated group Name {
    repeated group Language {
      required string Code;
      optional string Country; }
    optional string Url; }}
```

DocId: 20	r₂
Links	
Backward: 10	
Backward: 30	
Forward: 80	
Name	
Url: 'http://C'	

Figure 7. Automaton for assembling records from two fields, and the records it produces.

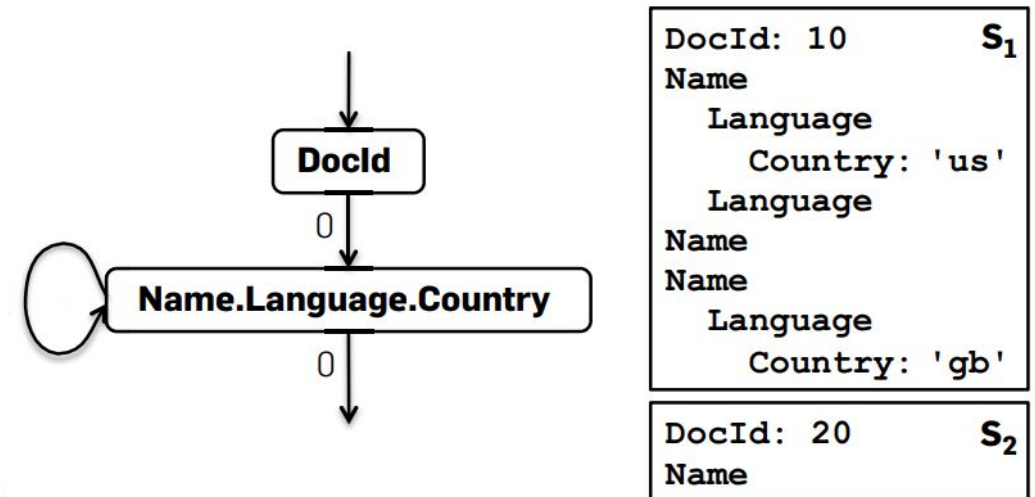


Figure 3. Column-striped representation of the data in Figure 2.

DocId	Name.Url	Links.Forward	Links.Backward
value r d	value r d	value r d	value r d
10 0 0	http://A 0 2	20 0 2	NULL 0 1
20 0 0	http://B 1 2	40 1 2	10 0 2
	NULL 1 1	60 1 2	30 1 2
	http://C 0 2	80 0 2	

Name.Language.Code	Name.Language.Country
value r d	value r d
en-us 0 2	us 0 3
en 2 2	NULL 2 2
NULL 1 1	NULL 1 1
en-gb 1 2	gb 1 3
NULL 0 1	NULL 0 1

Source: Dremel paper

HS23 Exam Q28, Q30 & Q31

Long examples, best to see on your own device.



<https://exams.vis.ethz.ch/exams/tx1ugdus.pdf>

HS23 Exam Q28: XML Validation

Task: Identify which schemas validate given XML document:
(Note: Assume bookType matches given books)

```
<library>
  <book>
    <title value="Data Structures and Algorithms"/>
    <author value="Jane Doe"/>
    <genre>Fiction</genre>
    <year>2020</year>
    <publisher>ABC Publishing</publisher>
  </book>
  <book>
    <title value="The History of Mathematics"/>
    <author value="John Smith"/>
    <genre>Non-Fiction</genre>
    <year>2018</year>
    <publisher>XYZ Publishing</publisher>
    <awardWinner/>
  </book>
</library>
```

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="library">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="book" minOccurs="2" maxOccurs="unbounded" type="bookType"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Valid.

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="library" type="xs:anyType"/>
</xs:schema>
```

Valid.

HS23 Exam Q28: XML Validation, Cont'd

Task: Identify which schemas validate given XML document:
(Note: Assume bookType matches given books)

```
<library>
  <book>
    <title value="Data Structures and Algorithms"/>
    <author value="Jane Doe"/>
    <genre>Fiction</genre>
    <year>2020</year>
    <publisher>ABC Publishing</publisher>
  </book>
  <book>
    <title value="The History of Mathematics"/>
    <author value="John Smith"/>
    <genre>Non-Fiction</genre>
    <year>2018</year>
    <publisher>XYZ Publishing</publisher>
    <awardWinner/>
  </book>
</library>
```

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="library">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="book" maxOccurs="unbounded">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="title" type="xs:string"/>
              <xs:element name="author" type="xs:string"/>
              <xs:element name="genre" type="xs:string"/>
              <xs:element name="year" type="xs:string"/>
              <xs:element name="publisher" type="xs:string"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Invalid. (*awardWinner* isn't allowed.)

HS23 Exam Q28: XML Validation, Cont'd

Task: Identify which schemas validate given XML document:
(Note: Assume bookType matches given books)

```
<library>
  <book>
    <title value="Data Structures and Algorithms"/>
    <author value="Jane Doe"/>
    <genre>Fiction</genre>
    <year>2020</year>
    <publisher>ABC Publishing</publisher>
  </book>
  <book>
    <title value="The History of Mathematics"/>
    <author value="John Smith"/>
    <genre>Non-Fiction</genre>
    <year>2018</year>
    <publisher>XYZ Publishing</publisher>
    <awardWinner/>
  </book>
</library>
```

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="library">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="book" maxOccurs="1" type="bookType"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Invalid. (Violate *maxOccurs*)

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="library" type="xs:string"/>
</xs:schema>
```

Invalid. (library is not a *string*)

HS23 Exam Q30: JSON Schema

```
{
  "type": "object",
  "properties": {
    "director": {
      "type": "object",
      "description": "Information of the movie director",
      "properties": {
        "name": {
          "type": "string"
        },
        "age": {
          "type": "integer",
          "minimum": 25
        },
        "nationalities": {
          "type": "array",
          "uniqueItems": true,
          "minItems": 1,
          "items": {
            "type": "string"
          }
        }
      }
    }
  }
},
```

```
    "movie": {
      "type": "object",
      "description": "Information of the movie",
      "properties": {
        "title": {
          "type": "string"
        },
        "producer": {
          "type": ["string", "null"]
        },
        "cast": {
          "type": "array",
          "items": {
            "type": "object",
            "properties": {
              "actor": {
                "type": "string"
              },
              "role": {
                "type": "string"
              }
            }
          },
          "minItems": 1
        }
      }
    }
  },
  "required": ["director", "movie"],
  "additionalProperties": false
}
```

```
{
  "director": {
    "name": "Martin Scorsese",
    "age": 78
  },
  "movie": {
    "title": "Goodfellas",
    "cast": [
      {
        "actor": "Robert De Niro",
        "role": "Jimmy Conway"
      }
    ]
  }
}
```

Valid.

HS23 Exam Q30: JSON Schema, Cont'd

```
{
  "type": "object",
  "properties": {
    "director": {
      "type": "object",
      "description": "Information of the movie director",
      "properties": {
        "name": {
          "type": "string"
        },
        "age": {
          "type": "integer",
          "minimum": 25
        },
        "nationalities": {
          "type": "array",
          "uniqueItems": true,
          "minItems": 1,
          "items": {
            "type": "string"
          }
        }
      }
    }
  }
},
```

```
    "movie": {
      "type": "object",
      "description": "Information of the movie",
      "properties": {
        "title": {
          "type": "string"
        },
        "producer": {
          "type": ["string", "null"]
        },
        "cast": {
          "type": "array",
          "items": {
            "type": "object",
            "properties": {
              "actor": {
                "type": "string"
              },
              "role": {
                "type": "string"
              }
            }
          },
          "minItems": 1
        }
      }
    }
  },
  "required": ["director", "movie"],
  "additionalProperties": false
}
```

```
{
  "director": {
    "name": "Christopher Nolan",
    "age": 50,
    "nationalities": [ "British" ]
  },
  "movie": {
    "title": "Inception",
    "producer": "Emma Thomas",
    "cast": [
      {
        "actor": "Leonardo DiCaprio",
        "role": "Cobb"
      }
    ]
  }
}
```

Valid.

HS23 Exam Q30: JSON Schema, Cont'd

```
{
  "type": "object",
  "properties": {
    "director": {
      "type": "object",
      "description": "Information of the movie director",
      "properties": {
        "name": {
          "type": "string"
        },
        "age": {
          "type": "integer",
          "minimum": 25
        },
        "nationalities": {
          "type": "array",
          "uniqueItems": true,
          "minItems": 1,
          "items": {
            "type": "string"
          }
        }
      }
    }
  }
},
```

```
    "movie": {
      "type": "object",
      "description": "Information of the movie",
      "properties": {
        "title": {
          "type": "string"
        },
        "producer": {
          "type": ["string", "null"]
        },
        "cast": {
          "type": "array",
          "items": {
            "type": "object",
            "properties": {
              "actor": {
                "type": "string"
              },
              "role": {
                "type": "string"
              }
            }
          },
          "minItems": 1
        }
      }
    }
  },
  "required": ["director", "movie"],
  "additionalProperties": false
}
```

```
{
  "director": {
    "name": "Quentin Tarantino",
    "age": 30,
    "nationalities": ["American"]
  },
  "movie": {
    "title": "Pulp Fiction",
    "producer": "Lawrence Bender",
    "cast": []
  }
}
```

Invalid.

(cast's *minItems*.)

HS23 Exam Q30: JSON Schema, Cont'd

```
{
  "type": "object",
  "properties": {
    "director": {
      "type": "object",
      "description": "Information of the movie director",
      "properties": {
        "name": {
          "type": "string"
        },
        "age": {
          "type": "integer",
          "minimum": 25
        },
        "nationalities": {
          "type": "array",
          "uniqueItems": true,
          "minItems": 1,
          "items": {
            "type": "string"
          }
        }
      }
    }
  }
},
```

```
    "movie": {
      "type": "object",
      "description": "Information of the movie",
      "properties": {
        "title": {
          "type": "string"
        },
        "producer": {
          "type": ["string", "null"]
        },
        "cast": {
          "type": "array",
          "items": {
            "type": "object",
            "properties": {
              "actor": {
                "type": "string"
              },
              "role": {
                "type": "string"
              }
            }
          },
          "minItems": 1
        }
      }
    }
  },
  "required": ["director", "movie"],
  "additionalProperties": false
}
```

```
{
  "director": {
    "name": "Steven Spielberg",
    "nationalities": ["American"]
  },
  "movie": {
    "title": "Jurassic Park",
    "producer": null,
    "cast": [
      {
        "actor": "Sam Neill",
        "role": "Dr. Alan Grant"
      },
      {
        "actor": "Laura Dern",
        "role": "Dr. Ellie Sattler"
      }
    ]
  },
  "awardNominations": 3
}
```

Invalid. (*awardNominations* aren't allowed)

HS23 Exam Q31: JSound

Determine for each of the JSound schemas, whether the document is valid or invalid against it.

```
{
  "bookTitle": "The Great Gatsby",
  "author": {
    "firstName": "F. Scott",
    "lastName": "Fitzgerald"
  },
  "publishedYear": 1925,
  "genres": ["Novel", "Fiction"],
  "availableFormats": ["Hardcover", "Paperback", "eBook"],
  "inStock": true
}
```

Invalid.

(Key presence is optional,
but *availableFormats* is an array.)

```
{
  "bookTitle": "string",
  "author": {
    "firstName": "string",
    "lastName": "string"
  },
  "publishedYear": "integer",
  "genres": ["string"],
  "availableFormats": ["string"],
  "inStock": "boolean",
  "!numPages": "integer"
}
```

Invalid.

(Missing *numPages*.)

```
{
  "bookTitle": "string",
  "author": {
    "firstName": "string",
    "lastName": "string",
    "middleName": "string"
  },
  "publishedYear": "integer",
  "genres": ["string"],
  "availableFormats": "string",
  "inStock": "boolean"
}
```


HS23 Exam Q31: JSound, Cont'd

Determine for each of the JSound schemas, whether the document is valid or invalid against it.

```
{
  "bookTitle": "The Great Gatsby",
  "author": {
    "firstName": "F. Scott",
    "lastName": "Fitzgerald"
  },
  "publishedYear": 1925,
  "genres": ["Novel", "Fiction"],
  "availableFormats": ["Hardcover", "Paperback", "eBook"],
  "inStock": true
}
```

Valid.

```
{
  "bookTitle": "string",
  "author": {
    "firstName": "string",
    "lastName": "string"
  },
  "publishedYear": "integer",
  "genres": ["string"],
  "inStock": "boolean"
}
```

Invalid.
(Closed
object types.)

```
{
  "bookTitle": "string",
  "author": {
    "firstName": "string",
    "lastName": "string"
  },
  "publishedYear": "integer",
  "genres": ["string"],
  "availableFormats": ["string"],
  "inStock": "boolean"
}
```

ETH zürich

See you next week!

Aljaž Medič
amedic@ethz.ch



Slides



Suggestions