Einsatz und Realisierung von Datenbanksystemen

ERDB Übungsleitung

Alice Rey, Maximilian {Bandle, Schüle}, Michael Jungmair

i3erdb@in.tum.de

Folien erstellt von Maximilian Bandle & Alexander Beischl

Organisatorisches
Disclaimer

Die Folien werden von der Übungsleitung allen Tutoren zur Verfügung gestellt.

Sollte es Unstimmigkeiten zu den Vorlesungsfolien von Prof. Kemper geben, so sind die Folien aus der Vorlesung ausschlaggebend.

Falls Ihr einen Fehler oder eine Unstimmigkeit findet, schreibt an i3erdb@in.tum.de mit Angabe der Foliennummer.

XML-Anfragesprachen

XML-Anfragesprachen
XQuery

Basiert auf XPath und kombiniert Ergebnisse der Anfragen

FLWOR-Syntax

- **For** Schleifen
- **Let** Variablen definieren
- **Where** Selektieren
- **Order By** Sortieren
- **Return** Ergebnis als neues XML formatieren

XML-Anfragesprachen
XQuery

Es muss nicht die komplette FLWOR Syntax genutzt werden, aber immer wenn FLW oder O genutzt werden, braucht man return

Variablen dürfen XML oder Unterabfragen (XPath oder XQuery) enthalten
Alle Variablen beginnen mit $

Beim Einbetten von XQuery in XML müssen geschweifte Klammern benutzt werden (und auch nur dann)
<XML>{XQuery}</XML>

XML-Anfragesprachen
XQuery-Beispielsanfrage

```
<Professoren>
  {
    for $p in doc('uni2')//ProfessorIn
      let $v := $p/Vorlesungen/Vorlesung
      where count ($v) > 0
      order by sum ($v/SWS)
      return
        <ProfessorIn>
          {$p/Name}
          <Belastung>{sum($v/SWS)}</Belastung>
        </ProfessorIn>
  }
</Professoren>
```

XML-Anfragesprachen
XQuery Ergebnis

```
<Professoren>
  {
    for $p in doc('uni2')//ProfessorIn
      let $v := $p/Vorlesungen/Vorlesung
      where count ($v) > 0
      order by sum ($v/SWS)
      return
        <ProfessorIn>
          {$p/Name}
          <Belastung>{sum($v/SWS)}</Belastung>
        </ProfessorIn>
  }
</Professoren>
```

Ausgabe:

```
<Professoren>
  <ProfessorIn>
    <Name>Russel</Name>
    <Belastung>6</Belastung>
  </ProfessorIn>
  <ProfessorIn>
    <Name>Sokrates</Name>
    <Belastung>10</Belastung>
  </ProfessorIn>
  <ProfessorIn>
    <Name>Kant</Name>
    <Belastung>10</Belastung>
  </ProfessorIn>
</Professoren>
```

Hinweise Die Aufgaben können auf `http://xquery.db.in.tum.de/` getestet werden. Die Daten für das Unischema können mit doc('uni2') geladen werden. Zur Lösung der Aufgaben können sie die folgenden XQuery-Funktionen verwenden:

max(NUM), count(X), tokenize(STR,SEP), sum(NUM), contains(HAY,NEEDLE)

1. max(NUMBERS) - Returns largest number from list
2. count(LIST) - Return the number of elements in the list
3. tokenize(STR,SEP) - Splits up the string at the seperator
4. sum(NUMBERS) - Returns sum of all numbers in list
5. contains(HAY,NEEDLE) - Checks if the search string (NEEDLE) is contained in the string (HAY)
6. distinct-values(LIST) - Returns the distinct values from the list

Aufgabe 1

Lösen Sie mit XQuery folgende Anfragen und testen Sie diese auf `xquery.db.in.tum.de`.

1. Geben Sie eine nach Rang sortierte Liste der Professoren aus (C4 oben).
2. Finden Sie die Namen der Professoren, die die meisten Assistenten haben.
3. Finden Sie für jede von einem Student gehörte Prüfung den Namen des Prüfers und Vorlesung.

Aufgabe 1

```
<Universitaet UnivName="Virtuelle Universitaet …“>
 <UniLeitung>… </UniLeitung>
 <Fakultaeten>
  <Fakultaet>
   <FakName>Theologie</FakName>
   <ProfessorIn ID="P2134" PersNr="P2134">
    <Name>Augustinus</Name>
    <Rang>C3</Rang>
    <Raum>309</Raum>
    <Vorlesungen>
     <Vorlesung ID="V5022" VorlNr="V5022">
      <Titel>Glaube und Wissen</Titel>
      <SWS>2</SWS>
     </Vorlesung>
    </Vorlesungen>
    <Assistenten>
     <Assistent ID="P3007" PersNr="P3007">
      <Name>Spinoza</Name>
      <Fachgebiet>Gott und Natur</Fachgebiet>
     </Assistent>
    </Assistenten>
   </ProfessorIn>
  </Fakultaet>
        ...
 </Fakultaeten>
 <Studenten>
  <Student ID="M24002" MatrNr="M24002">
   <Name>Xenokrates</Name>
   <Semester>18</Semester>
  </Student>
  <Student ID="M25403" MatrNr="M25403">
   <Name>Jonas</Name>
   <Semester>12</Semester>
   <hoert Vorlesungen="V5022"/>
   <Pruefungen>
    <Pruefung Pruefer="P2125" Vorlesung="V5041"
              Note="2.0"/>
   </Pruefungen>
  </Student>
        ...
  <Student ID="M1337" MatrNr="M1337">
   <Name>1337</Name>
   <Semester>9</Semester>
   <hoert Vorlesungen="V5022 V5041 … V4630"/>
  </Student>
 </Studenten>
</Universitaet>
```

1. Geben Sie eine nach Rang sortierte Liste der Professoren aus (C4 oben).

Aufgabe 1

```
<Universitaet UnivName="Virtuelle Universitaet …“>
 <UniLeitung>… </UniLeitung>
 <Fakultaeten>
  <Fakultaet>
   <FakName>Theologie</FakName>
   <ProfessorIn ID="P2134" PersNr="P2134">
    <Name>Augustinus</Name>
    <Rang>C3</Rang>
    <Raum>309</Raum>
    <Vorlesungen>
     <Vorlesung ID="V5022" VorlNr="V5022">
      <Titel>Glaube und Wissen</Titel>
      <SWS>2</SWS>
     </Vorlesung>
    </Vorlesungen>
    <Assistenten>
     <Assistent ID="P3007" PersNr="P3007">
      <Name>Spinoza</Name>
      <Fachgebiet>Gott und Natur</Fachgebiet>
     </Assistent>
    </Assistenten>
   </ProfessorIn>
  </Fakultaet>
        ...
 </Fakultaeten>
 <Studenten>
  <Student ID="M24002" MatrNr="M24002">
   <Name>Xenokrates</Name>
   <Semester>18</Semester>
  </Student>
  <Student ID="M25403" MatrNr="M25403">
   <Name>Jonas</Name>
   <Semester>12</Semester>
   <hoert Vorlesungen="V5022"/>
   <Pruefungen>
    <Pruefung Pruefer="P2125" Vorlesung="V5041"
              Note="2.0"/>
   </Pruefungen>
  </Student>
        ...
  <Student ID="M1337" MatrNr="M1337">
   <Name>1337</Name>
   <Semester>9</Semester>
   <hoert Vorlesungen="V5022 V5041 … V4630"/>
  </Student>
 </Studenten>
</Universitaet>
```

2. Finden Sie die Namen der Professoren, die die meisten Assistenten haben.

Aufgabe 1

```
<Universitaet UnivName="Virtuelle Universitaet …“>
 <UniLeitung>… </UniLeitung>
 <Fakultaeten>
  <Fakultaet>
   <FakName>Theologie</FakName>
   <ProfessorIn ID="P2134" PersNr="P2134">
    <Name>Augustinus</Name>
    <Rang>C3</Rang>
    <Raum>309</Raum>
    <Vorlesungen>
     <Vorlesung ID="V5022" VorlNr="V5022">
      <Titel>Glaube und Wissen</Titel>
      <SWS>2</SWS>
     </Vorlesung>
    </Vorlesungen>
    <Assistenten>
     <Assistent ID="P3007" PersNr="P3007">
      <Name>Spinoza</Name>
      <Fachgebiet>Gott und Natur</Fachgebiet>
     </Assistent>
    </Assistenten>
   </ProfessorIn>
  </Fakultaet>
        ...
 </Fakultaeten>
 <Studenten>
  <Student ID="M24002" MatrNr="M24002">
   <Name>Xenokrates</Name>
   <Semester>18</Semester>
  </Student>
  <Student ID="M25403" MatrNr="M25403">
   <Name>Jonas</Name>
   <Semester>12</Semester>
   <hoert Vorlesungen="V5022"/>
   <Pruefungen>
    <Pruefung Pruefer="P2125" Vorlesung="V5041"
              Note="2.0"/>
   </Pruefungen>
  </Student>
        ...
  <Student ID="M1337" MatrNr="M1337">
   <Name>1337</Name>
   <Semester>9</Semester>
   <hoert Vorlesungen="V5022 V5041 … V4630"/>
  </Student>
 </Studenten>
</Universitaet>
```

3. Finden Sie für jede von einem Student gehörte Prüfung den Namen des Prüfers und Vorlesung.

Aufgabe 2

Geben Sie ein Vorlesungsverzeichnis aus, welches nach dem Umfang der Vorlesungen in SWS gruppiert ist [1].

Die Ausgabe Ihrer Anfrage soll wie folgt aufgebaut sein:

```
<Vorlesungsverzeichnis>
  <Vorlesungen SWS="2">
    <Vorlesung VorlNr="V5216" Titel="Bioethik"/>
    <Vorlesung VorlNr="V5259" Titel="Der Wiener Kreis"/>
    <Vorlesung VorlNr="V5022" Titel="Glaube und Wissen"/>
    <Vorlesung VorlNr="V5049" Titel="Maeeutik"/>
  </Vorlesungen>
  <Vorlesungen SWS="3">
    <Vorlesung VorlNr="V5043" Titel="Erkenntnistheorie"/>
    <Vorlesung VorlNr="V5052" Titel="Wissenschaftstheorie"/>
  </Vorlesungen>
  <Vorlesungen SWS="4">
    <Vorlesung VorlNr="V4630" Titel="Die 3 Kritiken"/>
    <Vorlesung VorlNr="V5041" Titel="Ethik"/>
    <Vorlesung VorlNr="V5001" Titel="Grundzuege"/>
    <Vorlesung VorlNr="V4052" Titel="Logik"/>
  </Vorlesungen>
</Vorlesungsverzeichnis>
```

Aufgabe 3

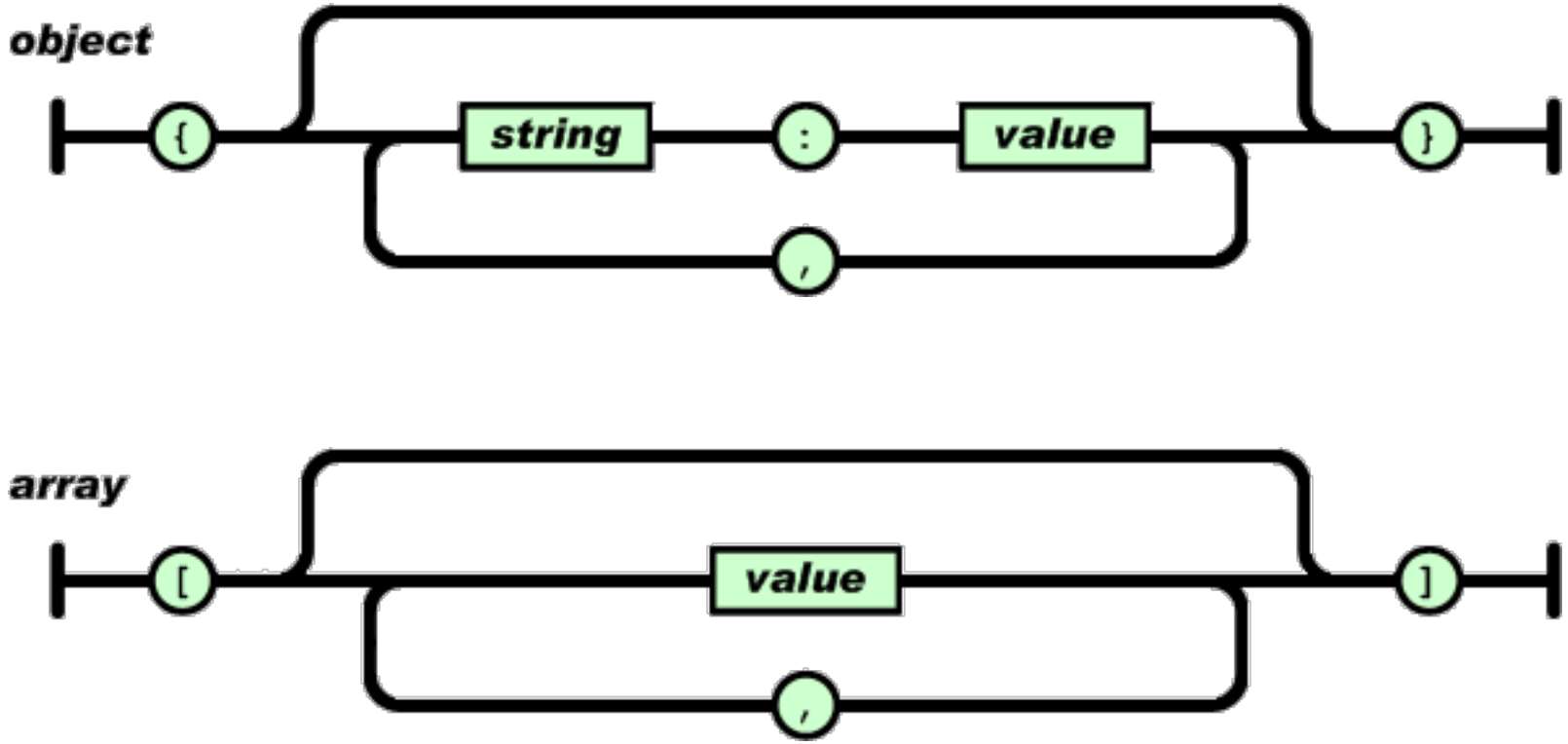
Schreiben Sie eine Anfrage, die folgendes zurück gibt:

```
<Universitaet>
  <Fakultaet Name="Philosophie" AnzahlAssistenten="3">
    <Professor Name="Sokrates" AnzahlAssistenten="2"/>
    <Professor Name="Russel" AnzahlAssistenten="1"/>
  </Fakultaet>
  <Fakultaet Name="Physik" AnzahlAssistenten="2">
    <Professor Name="Kopernikus" AnzahlAssistenten="2"/>
  </Fakultaet>
  <Fakultaet Name="Theologie" AnzahlAssistenten="1">
    <Professor Name="Augustinus" AnzahlAssistenten="1"/>
  </Fakultaet>
</Universitaet>
```

JSON

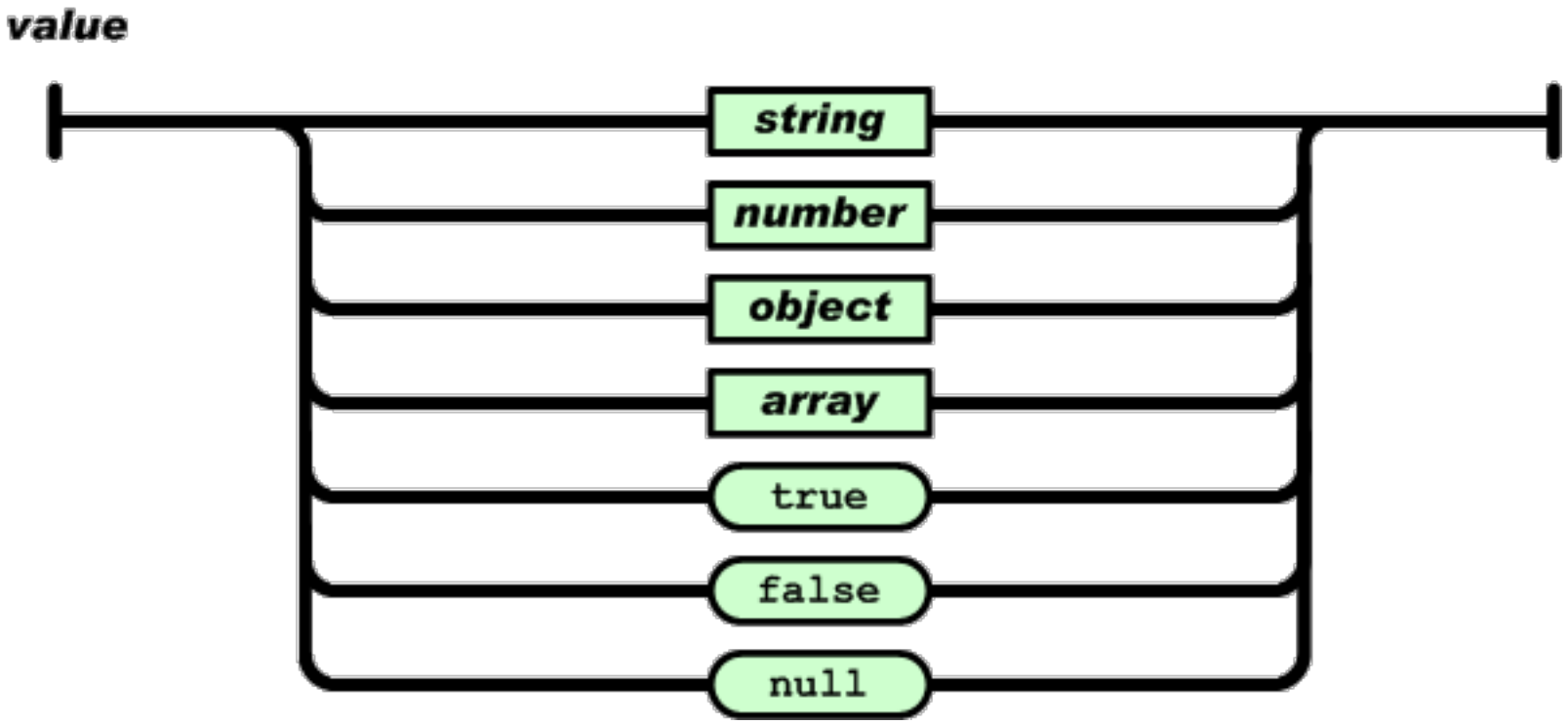
- JSON baut auf zwei Strukturen auf:
  - Objekt
  - Array

JSON



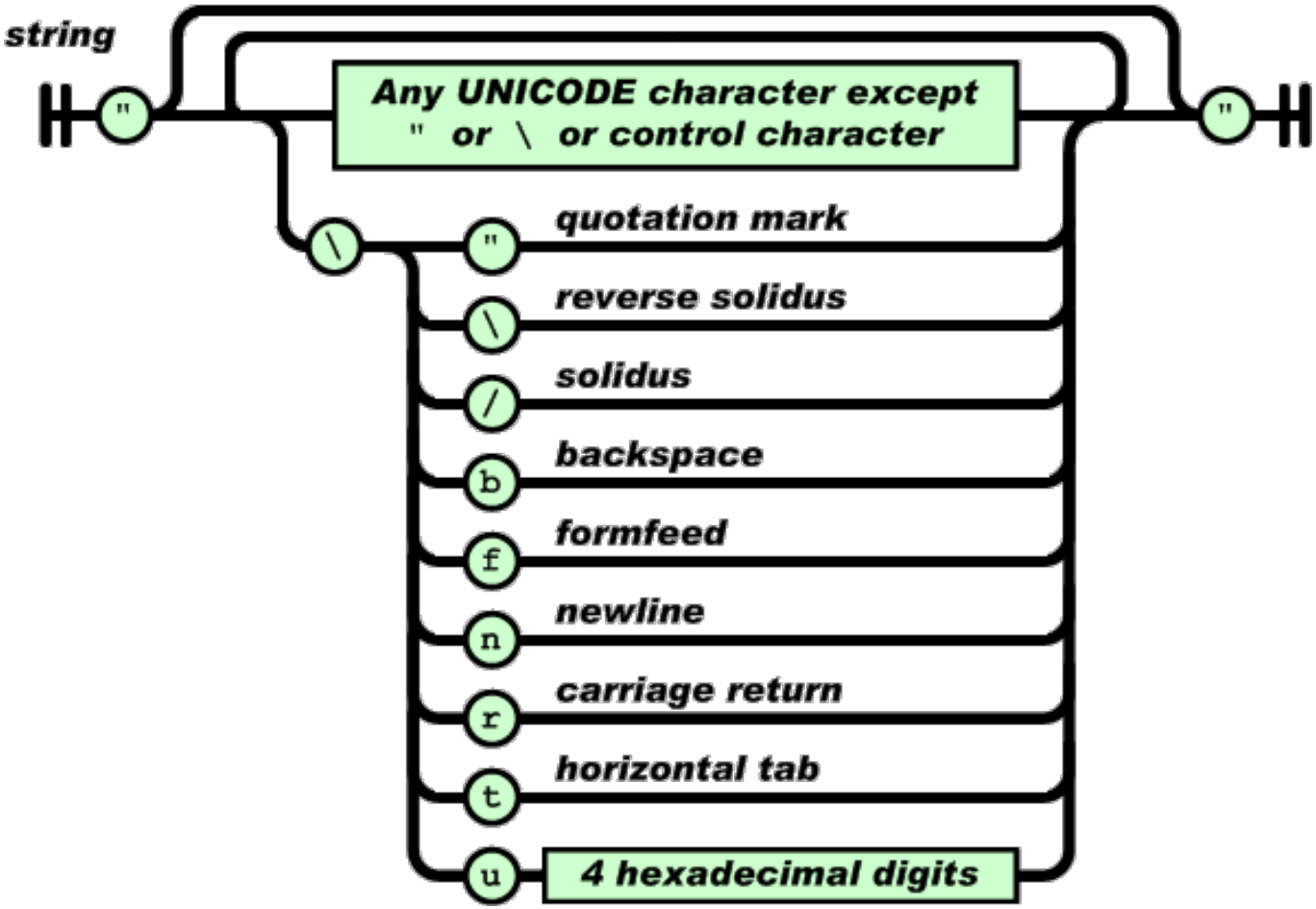
Grafiken von: https://www.json.org

JSON



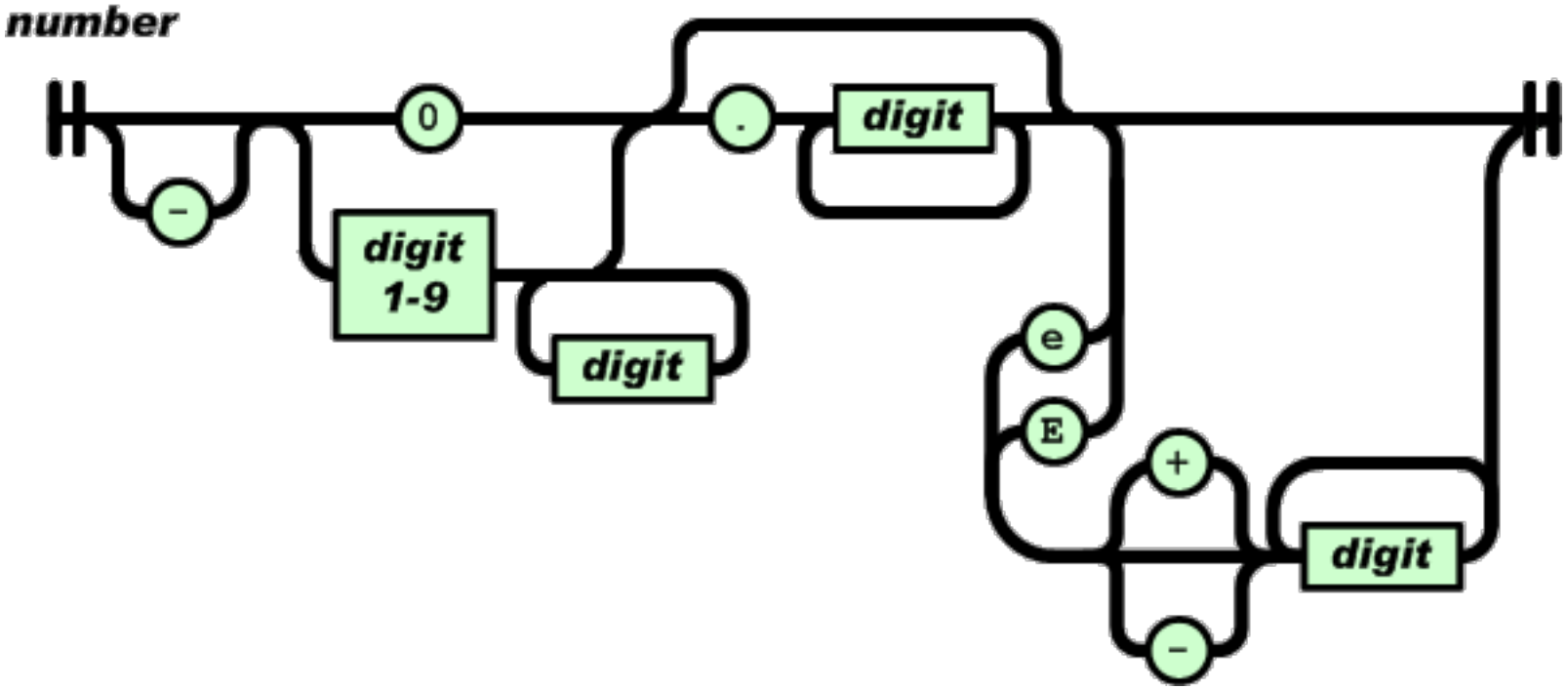
Grafiken von: https://www.json.org

JSON



Grafiken von: https://www.json.org

JSON



Grafiken von: https://www.json.org

Gruppenaufgabe 4

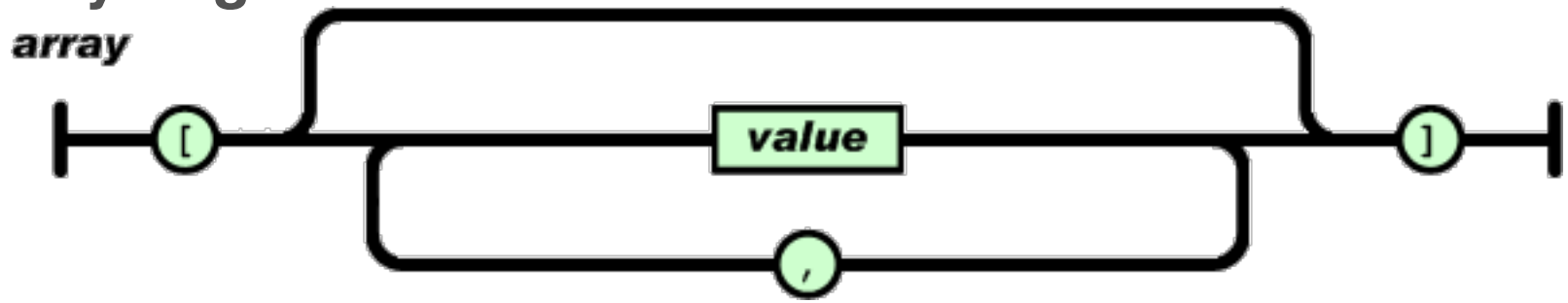
Überlegen Sie sich, wie Ihre Visitenkarte im JSON-Format aussähe und stellen Sie diese in der Übung vor.

JSON in SQL

- Seit 2017 ist JSON-Syntax zum Einbetten als ein Attribut in SQL standardisiert
- Zugriff erfolgt auf das Attribut als Ausdruck in der SELECT- oder WHERE-Klausel
- SQL stellt Funktionen bereit, um auf Teile des JSON-Attributes zuzugreifen

JSON in SQL

Array Zugriff



	Operand	Ergebnis	Beschreibung
->	Zahl	JSON	Element an angebener Position

Beispiel:

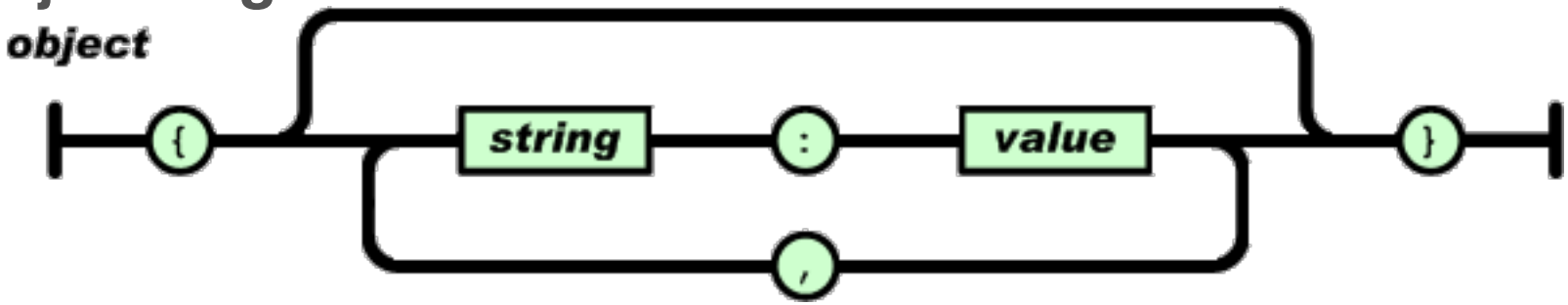
```
'[{"a":"foo"},{"b":"bar"},{"c":"baz"}]'::json->2
```

Ergebnis:

```
{“c":"baz"}
```

JSON in SQL

Objekt Zugriff



	Operand	Ergebnis	Beschreibung
->	Text	JSON	Element zu angebenem Schlüssel

Beispiel:

```
'{"a": {“b”:"foo"}, "b": {"c":"bar"}}'::json->'a'
```

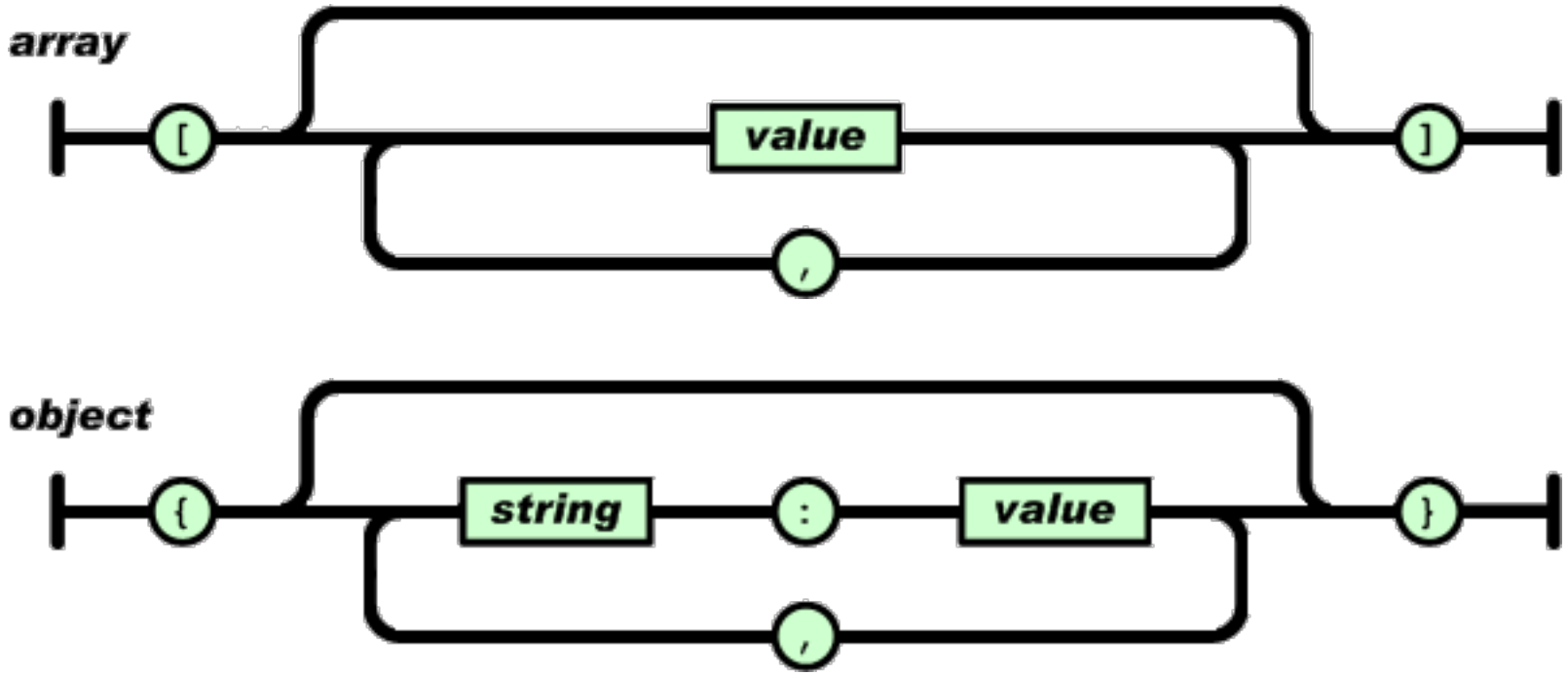
Ergebnis:

```
{"b":"foo"}
```

JSON in SQL
Navigation

JSON besteht aus geschachtelten Arrays und Objekten

=> Navigation durch Kombination aus Array- und Objektzugriffen



Grafiken von: https://www.json.org

JSON in SQL
Ausgabe als Text

Operator –> gibt immer JSON zurück.

=> Das Ergebnis lässt sich nicht mit anderen Werten in SQL vergleichen

```
select '["a","b","c"]'::json->1 = 'b';
```

Fehler: Cast failed - JSON und Text nicht vergleichbar

Lösung:

Operator –>> gibt das Ergebnis als Text zurück.

=> Ergebnis kann auch gecasted werden um z.B. mit Zahl zu vergleichen

```
select '["a","b","c"]'::json->>1 = 'b';
```

Ergebnis: True

Grafiken von: https://www.json.org

Aufgabe 5

Datenbanksysteme erlauben JSON-Objekte eingebettet als Attribute in Tabellen. Der zugehörige Syntax ist seit 2017 standardisiert[2] und zum Beispiel in PostgreSQL integriert[3]. Das nachfolgende Statement erstellt eine Hilfstabelle, die einen Ausschnitt des Uni-Schemas als JSON-Objekt enthält (und lässt sich in `hyper-db.de` eingeben).

```
with uni_json (name, doc) as (values ('VirtU', '{
  "Name": "Virtuelle Universitaet der Grossen Denker",
  "UniLeitung": {"Rektor": "Sokrates", "Kanzler": "Erhard"},
  "Fakultaeten":  [
    { "Name": "Philosophie", "Professoren": [
        { "PersNr": 2125, "Name": "Sokrates", "Rang": "C4",
          "Vorlesungen": [ {"VorlNr": 5041, "Titel": "Ethik", "SWS": 4},
            {"VorlNr": 5049, "Titel": "Maeeutik", "SWS": 2},
            {"VorlNr": 4052, "Titel": "Logik", "SWS": 4}]
        }]}]}'::json))
```

1. Geben Sie in SQL den Namen der jeweils ersten Fakultät in `uni_json` aus.
2. Geben Sie in SQL die Personalnummer (`PersNr`) des ersten Professors der jeweils ersten Fakultät aus.
3. Joinen Sie diese mit der SQL-Relation `pruefen` und `Studenten`, um die Namen aller von ihm geprüften Studenten auszugeben.

Resource Description Framework (RDF)

- Semantisch reichhaltige Beschreibung der Web-Ressourcen
- Nutzt URIs (Uniform Resource Identifiers) um Entities zu identifizieren

- RDF Datenbasis besteht aus:

(Subjekt, Prädikat, Objekt)

- Leicht als Graph zu visualisieren

Resource Description Framework (RDF)

RDF-Beispiel

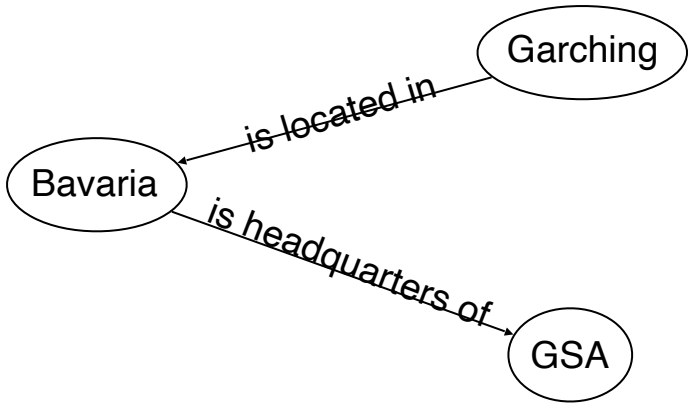
Garching is located in Bavaria.

Subject Predicate Object

Bavaria is headquarters of the German Ski Association.

Subject Predicate Object

Graph representation



SPO triplestore representation

Subject	*Predicate*	*Object*
Garching	is located in	Bavaria
Bavaria	is headquarters of	German Ski Association

Resource Description Framework (RDF)
SPARQL - Anfragesprache für RDF

- Finde alle Personen mit schwarzen Haaren und grünen Augen

```
SELECT ?n
WHERE {
  ?p rdf:type dbo:Person .
  ?p dbo:hairColor "Black" .
  ?p dbo:eyeColor "Green" .
  ?p dbp:name ?n .
}
```

Aufgabe 6

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
<#SOKRATES>
    a foaf:Person ;
    foaf:firstName "Sokrates" ;
    foaf:surName "Sokrates" ;
    foaf:name "Sokrates" ;
    foaf:age 30 ;
    foaf:knows [
        a foaf:Person ;
        foaf:name "Russel"
    ];
    foaf:knows [
        a foaf:Person ;
        foaf:name "MCurie"
    ] .
```

```
<#MCURIE>
    a foaf:Person ;
    foaf:firstName "Marie" ;
    foaf:name "MCurie" ;
    foaf:SurName "Curie" ;
    foaf:age 29 ;
    foaf:knows [
        a foaf:Person ;
        foaf:name "PCurie"
    ];
    foaf:knows [
        a foaf:Person ;
        foaf:name "Russel"
    ];
    foaf:knows [
        a foaf:Person ;
        foaf:name "Sokrates"
    ] .
```

```
<#PCURIE>
    a foaf:Person ;
    foaf:firstName "Pierre" ;
    foaf:name "PCurie" ;
    foaf:SurName "Curie" ;
    foaf:age 29 ;
    foaf:knows [
        a foaf:Person ;
        foaf:name "MCurie"
    ] .
<#RUSSEL>
    a foaf:Person ;
    foaf:firstName "Bertrand" ;
    foaf:name "Russel" ;
    foaf:SurName "Russel" ;
    foaf:age 97 ;
    foaf:knows [
        a foaf:Person ;
        foaf:name "Sokrates"
    ] .
```

Vervollständigen Sie die untere Anfrage um die Namen der Freunde von Personen mit dem Vornamen *Sokrates* zu finden, die älter als 30 Jahre sind. Die *foaf* Onthology is unter `http://xmlns.com/foaf/spec/` beschrieben. Nutzen Sie `https://rdf.db.in.tum.de/` für Ihre Abfrage.

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name2
WHERE {
        ....
    }
```

Aufgabe 7

```
@prefix ex: <http://example.org>.
ex:Rapunzel ex:hatAutor ex:Sokrates.
ex:Rapunzel ex:erschienen 2006.
ex:Aschenputtel ex:hatAutor ex:Archimedes.
ex:Aschenputtel ex:hatAutor ex:Platon.
ex:Schneewittchen ex:hatAutor ex:Platon.
ex:Schneewittchen ex:erschienen 2004.
```

Drücken Sie die folgenden Anfragen in SPARQL aus:

1. Geben Sie alle Bücher aus, für die sowohl der Autor als auch das Erscheinungsjahr in der Datenbank enthalten sind.
2. Geben Sie die gemeinsamen Autoren der beiden Bücher Aschenputtel und Schneewitchen aus.
3. Geben Sie die Namen aller Autoren (ohne Duplikate) von Büchern mit einem Erscheinungsjahr nach 2004 aus.

Fragen?