

Datenorientierte Systemanalyse

19/05 / 2014

Axel Polleres

Populäre Datenformate

Stundenwiederholung:

- **Stundenwiederholung:**
 - Überlegen Sie sich ein **eigenes** praktisches Datenbank-Schema und Erstellen Sie ein ER-Modell für Ihre Datenbank
 - Verwenden Sie PRIMARY KEY constraints und REFERENCES constraints richtig
 - Formulieren Sie SQL-Abfragen über Ihre Datenbank:
 - Mind. eine Abfrage, die mehrere Tabellen verknüpft (Join)
 - Filtern von Datensätzen nach verschiedenen Kriterien (WHERE)
 - Mind. eine UNION query
 - Mind. eine Abfrage, die Sortierung und Aggregation verwendet (ORDER BY, GROUP BY, COUNT, AVG, SUM...)

CSV/TSV

CSV = Comma-separated-Values ... kennen wir schon & können wir leicht in Postgres importieren (mit \COPY)

- Sehr verbreitet, v.a. auch im "Open Data" Bereich
- Probleme: viele CSVs entsprechen eigentlich nicht dem Standard:
<http://tools.ietf.org/html/rfc4180>
 - encoding-Unterschiede für Sonderzeichen, ';' oder ' ' als Trennzeichen, unterschiedliches encoding für end-of-line, ',' vs. '.' als Decimal-Trennzeichen, etc.
 - Eine neue Standardisierungsgruppe des W3C versucht diese Probleme zu adressieren:
<http://www.w3.org/2013/csvw/>

TSV ... sehr aehnlich zu CSV, [TAB] als Trennzeichen, meist wengier "Konflikte" mit anderen Zeichen
<http://www.iana.org/assignments/media-types/text/tab-separated-values>

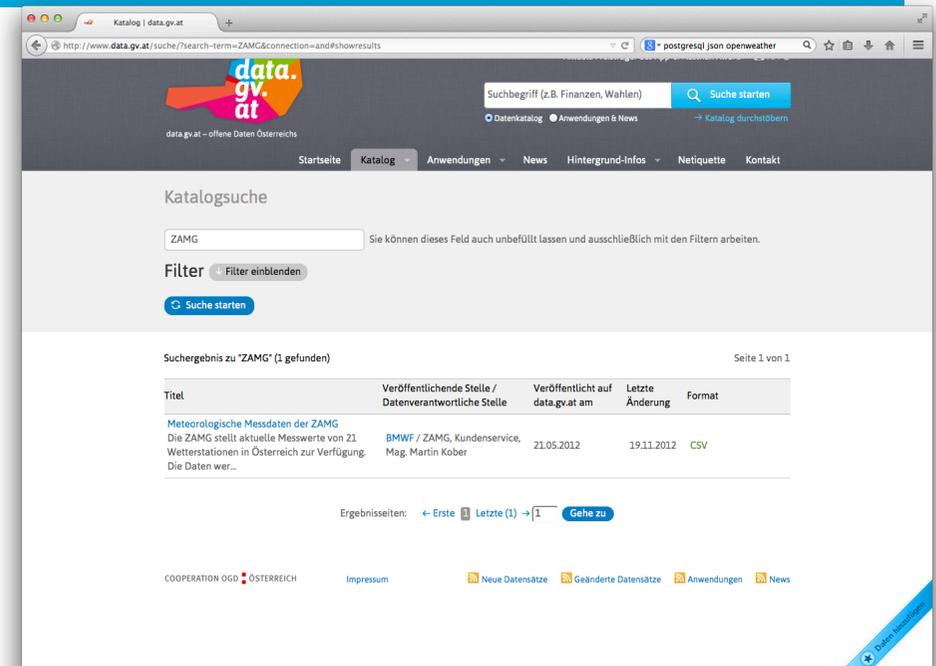
Import von TSV in Postgres geht genauso wieder mit COPY, z.B.

```
\COPY grades_raw FROM 'GradesDemo.tsv' WITH DELIMITER '\t' CSV HEADER
```

Viele CSVs in Open Data:

- <http://data.gv.at/>

- z.B.



- <http://www.zamg.ac.at/ogd/> → Wetterdaten von 20 Wetterstationen in Österreich.

JSON

JSON

JavaScript Object Notation

Was ist JSON?

JSON ist ein kompaktes, textbasiertes und für Menschen als auch Computer leicht les-/parsebares Datenformat, welches für den einfachen Daten- und Informationsaustausch zwischen Applikationen entwickelt wurde.

JSON ist ein Datenformat und keine (Markup-)Sprache!



1. Es ist **kompakt, textbasiert** und **leicht lesbar und leicht zu parsen**.
2. Es ist nicht proprietär/offen
3. Es **basiert auf** einem Subset von **JavaScript**.
4. Es ist **einfach zu verstehen, zu verändern** und **zu generieren**.

Warum JSON und nicht XML?

-  JSON ist kompakter als XML.
-  JSON Objects sind getyped (string, number, arrays, boolean), im Gegensatz zu XML wo alle Daten als Strings repräsentiert werden.
-  Daten im JSON Format sind sofort als JSON Objects in JavaScript verfügbar, ohne vorher übersetzt werden zu müssen wie es etwa bei XML der Fall wäre.

JSON Object Syntax

- Ein **ungeordnetes** Set von Attribut-Namen und deren Werten.
- Ein JSON Object beginnt immer mit einem '{' und endet mit einem '}'.
- Jedem Namen folgt ein ':'
- Namen/Wert Paare werden durch Beistriche voneinander getrennt.
- Nesting: JSON Object können verschachtelt sein, d.h. der Wert eines Attributs kann wieder ein Object sein.
- Arrays werden als geordnete Collection von Werten definiert und mittels [] umschlossen.

JSON Example

```
{  
  "first": "Jimmy",  
  "last": "James",  
  "age": 29,  
  "sex": "M",  
  "salary": 63000,  
  "department": {"id": 1, "name": "Sales"},  
  "registered": false,  
  "numbers": [ 2, 3, 11, 23 ],  
  "listofCustomers": [ {"name": "Customer1"},  
                       {"name": "Customer2"} ]  
}
```

JSON

JavaScript Object Notation

Seit PostgreSQL 9.3 ist JSON ein eigener Datentyp in postgres

<http://www.postgresql.org/docs/9.3/static/datatype-json.html>

<http://www.postgresql.org/docs/9.3/static/functions-json.html>

Verbinden Sie sich zu Ihrer PostgreSQL 9.3 Datenbank:

```
psql -h localhost -p 5454  
(user/pw wieder hMatNr)
```

JSON and PostgreSQL

Tabellen direkt mit JSON Daten speisen:

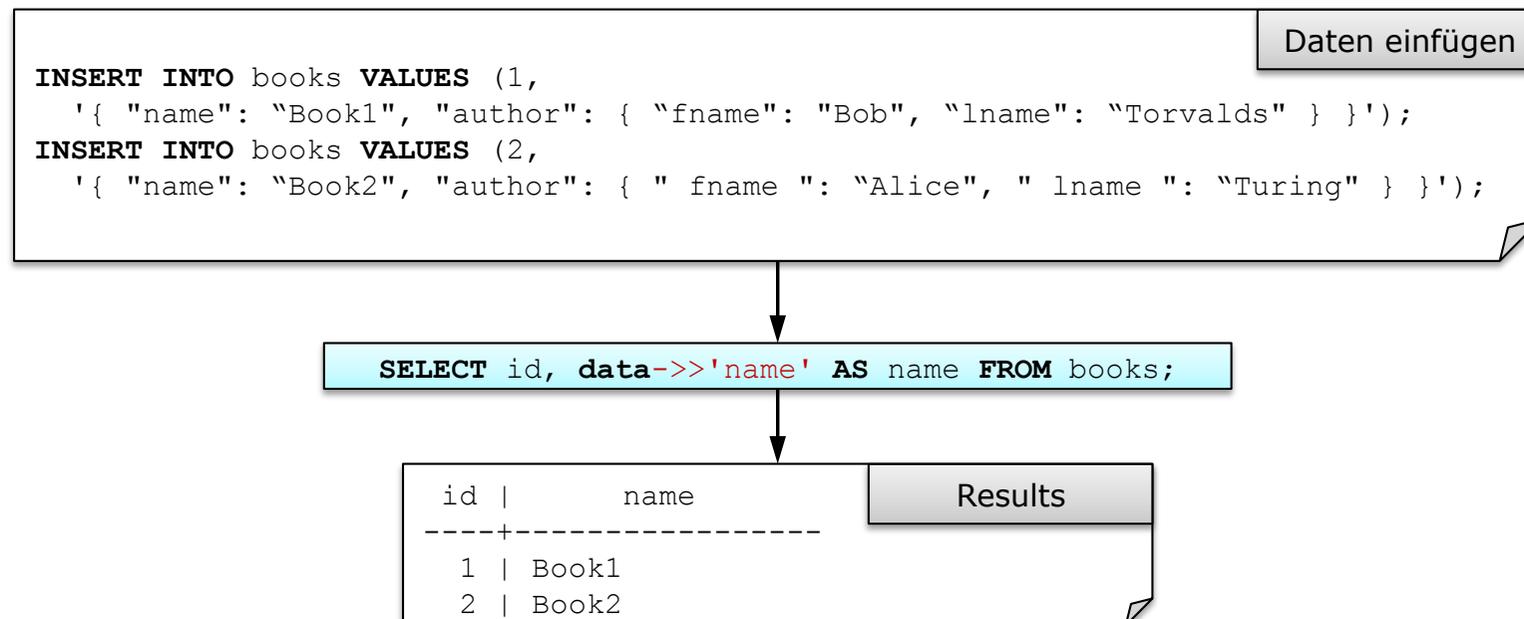
```
CREATE TABLE books ( id integer, data json );
```

Daten einfügen

```
INSERT INTO books VALUES (1,  
  '{ "name": "Book1", "author": { "fname": "Bob", "lname": "Torvalds" } }');  
INSERT INTO books VALUES (2,  
  '{ "name": "Book2", "author": { "fname": "Alice", "lname": "Turing" } }');  
INSERT INTO books VALUES (3,  
  '{ "name": "Book", "author": { "fname": "Hans", "lname": "Bauer" } }');  
INSERT INTO books VALUES (4,  
  '{ "name": "Book4", "author": { "fname": "Susi", "lname": "Huber" } }');
```

Selecting

Über das **data** Attribut kann auf die JSON Objects zugegriffen werden und mittels `->>` entweder eine reine Textrepräsentation erzwungen oder mit `->` der ursprüngliche JSON type zurückgeliefert werden (kann auch ein Objekt sein).



JSON

JavaScript Object Notation

Filtering

Man kann auch JSON Objects über die Werte ihrer genesteten Attribute finden.

```
INSERT INTO books VALUES (1,  
  '{ "name": "Book1", "author": { "fname": "Bob", "lname": "Torvalds" } }');  
INSERT INTO books VALUES (2,  
  '{ "name": "Book2", "author": { " fname ": "Alice", " lname ": "Turing" } }');
```

Daten einfügen

```
SELECT * FROM books WHERE data->'author'->'fname' = Alice';
```

```
id | data  
---+-----  
2 | '{ "name": "Book2", "author": { " fname ": "Alice", " lname ": "Turing" } }'
```

Results

JSON als output aus "normaler" Tabelle generieren:

- `SELECT row_to_json(t) from (SELECT LVnr || '/' || Semester as LV, count(Matnr) FROM Grades GROUP BY Lvnr, Semester) t;`
- '||' steht in SQL fuer string concatenation
 - Ein JSON-Array muessten Sie allerdings "von Hand" generieren, z.B. so:
 - `SELECT '[' || LVnr || '/' || Semester || ', ' || count(Matnr) || ']' FROM Grades GROUP BY Lvnr, Semester) t;`
 - Ein einfaches Hochkomma können Sie durch zwei aufeinanderfolgende " ausgeben:
`SELECT '[' || LVnr || '/' || Semester || ' ' || count(Matnr) || '], ' FROM Grades GROUP BY Lvnr, Semester;`
 - Mehr dazu nächste Stunde ...

JSON Praktische Beispiele

- JSON ist ein beliebtes API format:
- Wie bekomme ich Json Daten in meine Datenbank?
- Beispiele:
 - Open Weather Map <http://openweathermap.org/API>
 - Facebook Graph API
 - WU BACH API <https://bach.wu.ac.at/z/start/api>

Open Weather Map

- <http://api.openweathermap.org/data/2.5/weather?q=London,uk>

Tabelle anlegen um Wetterdaten zu importieren und abzufragen, siehe:

[https://ai.wu.ac.at/~polleres/teaching/DOSA_2014/20140519/
weatherdata_json.sql.txt](https://ai.wu.ac.at/~polleres/teaching/DOSA_2014/20140519/weatherdata_json.sql.txt)

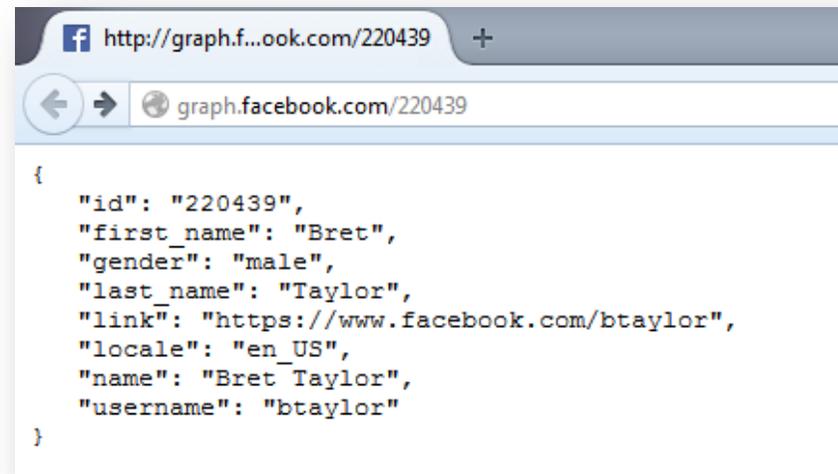
JSON

JavaScript Object Notation

JSON and FB Graph API

Mit Hilfe der FB Graph API können auch FB Inhalte mittels JSON geparsed bzw. ausgelesen werden.

Über <http://graph.facebook.com/<username>> oder <http://graph.facebook.com/<id>> kann auf die Basisinformationen zugegriffen werden.



```
{
  "id": "220439",
  "first_name": "Bret",
  "gender": "male",
  "last_name": "Taylor",
  "link": "https://www.facebook.com/btaylor",
  "locale": "en_US",
  "name": "Bret Taylor",
  "username": "btaylor"
}
```

JSON

JavaScript Object Notation

JSON and FB Graph API

Für „more sophisticated“ Abfragen muss ein Authentifizierungstoken erstellt werden. Dieser kann unter <https://developers.facebook.com/docs/graph-api/using-graph-api/v2.0> und einem Klick auf *Graph Explorer* angefordert werden.

- Entsprechende Permissions (z.b. likes) erteilen und „Get Access Token“ klicken
- Access token kopieren

The image shows a composite of two screenshots from the Facebook Graph API Explorer. The top screenshot, titled "Reading", explains that nodes and edges can be read with an HTTP GET request. A blue circle with the number "1" highlights the "Graph Explorer" link in the navigation bar. The bottom screenshot shows the main interface of the Graph API Explorer. A blue circle with the number "2" highlights the "Get App Token" button, and another blue circle with the number "3" highlights the "FQL Query" input field. The interface includes a navigation bar with "App: [?] Graph API Explorer", "Sprache: [?] Afrikaans", and "API-Version: [?] v2.0". The main area shows a "Zugriffsschlüssel" (Access Token) field with a value, a "Fehlerbehebung" (Troubleshooting) button, and a "Zugriffsschlüssel anfordern" (Request Access Token) button. Below this, there are tabs for "Graph API" and "FQL Query", a "GET" method selector, a query input field containing "/v2.0/me", and an "Absenden" (Send) button. A footer note says "Erfahre mehr über die Graph-API-Syntax."

JSON

JavaScript Object Notation

JSON and FB Graph API

Wurden die entsprechenden Berechtigungen erteilt, kann zb. auf den Newsfeed (`user_actions.news`), auf Freunde welche ebenfalls der Graph API Access Berechtigung erteilt haben (`user_friends`) und auf viele andere Informationen mittels JSON zugegriffen werden. Eine Reihe von *Common Usecases* kann <https://developers.facebook.com/docs/graph-api/common-scenarios/> gefunden werden.



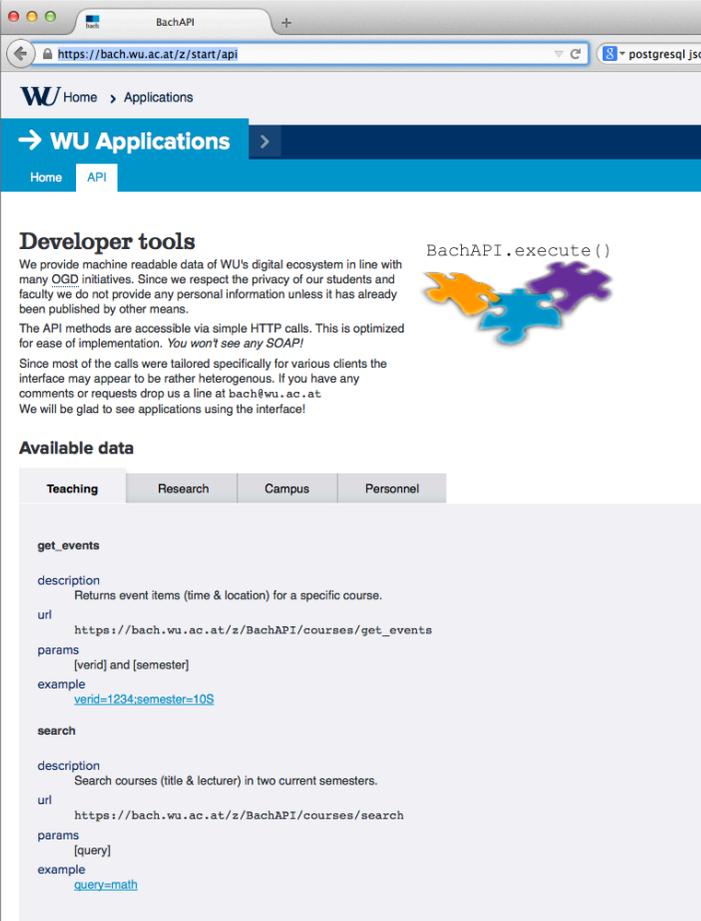
```
{
  "data": [
    {
      "name": "██████████",
      "id": "██████████"
    },
    {
      "name": "██████████████████",
      "id": "██████████"
    }
  ]
},
```

JSON from Facebook - Beispiel

- Extract and query your "Likes", check again the example at https://ai.wu.ac.at/~polleres/teaching/DOSA_2014/20140519/

Another example ... from WU! The BACH API

- <https://bach.wu.ac.at/z/start/api>
- Also allows to retrieve JSON, e.g. "courses by Prof. Taudes in WS12"
- ```
curl "https://bach.wu.ac.at/z/BachAPI/courses" \
-i -H 'Content-Type:application/json-rpc' \
--data '{"params": {"query": "taudes",
"semester": "12W"}, "id": "0815", "method": "search"}'
```



The screenshot shows a web browser window titled "BachAPI" with the URL <https://bach.wu.ac.at/z/start/api>. The page has a navigation bar with "Home" and "API" links. The main content area is titled "Developer tools" and includes a section for "Available data" with tabs for "Teaching", "Research", "Campus", and "Personnel". The "Teaching" tab is active, showing details for "get\_events" and "search" methods, including their descriptions, URLs, parameters, and example requests.

**Developer tools**

We provide machine readable data of WU's digital ecosystem in line with many OGD initiatives. Since we respect the privacy of our students and faculty we do not provide any personal information unless it has already been published by other means.

The API methods are accessible via simple HTTP calls. This is optimized for ease of implementation. You won't see any SOAP!

Since most of the calls were tailored specifically for various clients the interface may appear to be rather heterogenous. If you have any comments or requests drop us a line at [bach@wu.ac.at](mailto:bach@wu.ac.at). We will be glad to see applications using the interface!

`BachAPI.execute()`

**Available data**

| Teaching                                                                                                                                                                                                                                                                                                                                                                            | Research | Campus | Personnel |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------|--------|-----------|
| <p><b>get_events</b></p> <p><b>description</b><br/>Returns event items (time &amp; location) for a specific course.</p> <p><b>url</b><br/><a href="https://bach.wu.ac.at/z/BachAPI/courses/get_events">https://bach.wu.ac.at/z/BachAPI/courses/get_events</a></p> <p><b>params</b><br/>[verid] and [semester]</p> <p><b>example</b><br/><a href="#">verid=1234;semester=10S</a></p> |          |        |           |
| <p><b>search</b></p> <p><b>description</b><br/>Search courses (title &amp; lecturer) in two current semesters.</p> <p><b>url</b><br/><a href="https://bach.wu.ac.at/z/BachAPI/courses/search">https://bach.wu.ac.at/z/BachAPI/courses/search</a></p> <p><b>params</b><br/>[query]</p> <p><b>example</b><br/><a href="#">query=math</a></p>                                          |          |        |           |

## Other common formats:

- XML ... We have seen that already, (X)HTML is actually XML, XML is not too different from JSON, in that it also allows to encode nested, tree-shaped data:

```
{ "data": {
 "category": "University" ,
 "name": "WU Wien" } }
```

```
<data>
 <category>University</category>
 <name>WU Wien</name>
</data>
```

- XML has an own query languages: Xquery, XSLT
- RDF ... We will hear about that in one of the next lessons
  - RDF has an own query language: SPARQL

# Hausübung 3:

Importieren Sie JSON und CSV Daten in Ihre Datenbank!

1) Finden Sie interessante Datenquellen online und importieren Sie diese in Ihre Datenbank:

- z.B. ueber das Such-Interface von <http://data.gv.at> oder über das JSON API: <http://www.data.gv.at/katalog/api/>
  - Beispiel: Liste aller Datensätze:
    - <http://www.data.gv.at/katalog/api/search/dataset?limit=2000>
  - Meta-daten eines bestimmten Datensatzes:
    - <http://www.data.gv.at/katalog/api/rest/dataset/bev-lkerung-nach-geschlecht-und-staatsangeh-rigkeitsgruppen>
  - API Beschreibung:  
<http://ckan.readthedocs.org/en/ckan-1.7.1/api-v2.html>
- Anderes Beispiel: Europaisches Datenportal: <http://open-data.europa.eu/en/data/>
- Anderes Beispiel Liste von Web APIs: <http://www.programmableweb.com/>

2) Finden Sie Daten online, die zu Ihrem Datenbank-Schema von Hausübung 2 passen könnten:

<http://openflights.org/data.html> und importieren Sie diese (wenn möglich) in Ihre Datenbank. Wenn nicht, dann dokumentieren Sie zumindest, in welchem Format diese Daten vorliegen und wie Sie sie gerne in Ihre Datenbank importieren würden.

Ein Beispiel: <http://openflights.org/data.html>