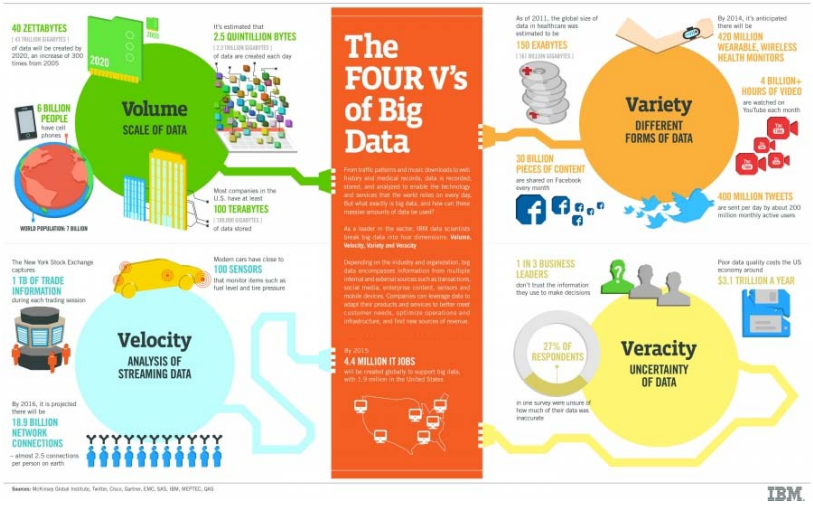
Schema-Extraktion und Structural Outlier Detection in JSON-based NoSQL Data Stores

Meike Klettke, Uta Störl, Stefanie Scherzinger
Universität Rostock, Hochschule Darmstadt, OTH Regensburg

Motivation

Schema-Management für NoSQL-Systeme

- NoSQL, verwendet zur Speicherung von
  - regulären, strukturierten und
  - irregulären, unstrukturierten Daten und
  - Objektspeicherung



- Häufig in „frühen Phasen“ der Entwicklung eingesetzt

Schema-Management in NoSQL-Datenbanken

Häufig:

- **Daten** in NoSQL-Datenbanksystemen sind **bereits vorhanden**
- Schema nur **implizit** in den Daten
- **Heterogenität** aufgrund
  - unterschiedlicher Daten in einer Kollektion (**Data Lake**) oder
  - verschiedene über die Zeit entstandene Versionen in einer Kollektion

Nachträgliche Schemaableitung

Fahrplan für den heutigen Vortrag

Verfahren zur Schema-Extraktion

I. Aufbau eines Structure Identification Graph (SG)
II. Ableitung des Schemas, Verfahren und Beispiele
III. Anwendung auf Datenbankstatistiken, Measures Ausreißererkennung
IV. Vereinfachung des Verfahrens (RG)
V. Umsetzung für mongodb und Messungen

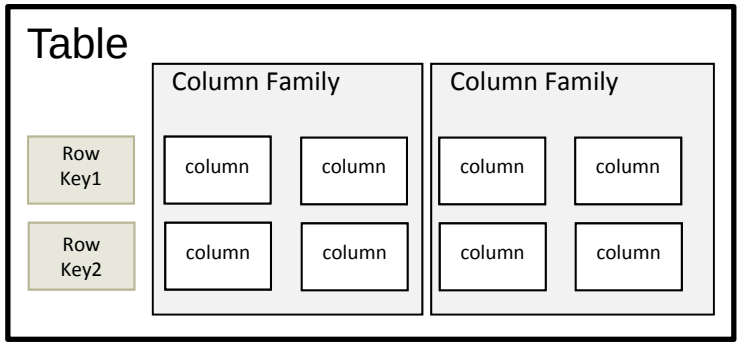
Klassifikation von NoSQL-Datenbanken

Große Vielfalt von Systemen, lassen sich in verschiedene Klassen unterteilen:

(1) Key-value stores

key	value
key	value
key	value

(2) **Extensible record stores / Column Family Databases**



(3) **Document stores (JSON)**

```
{  "id": 1,
   “property1": “value",
   “property2":  {
      “property3": “value”,
      “property4": “value” } }
```

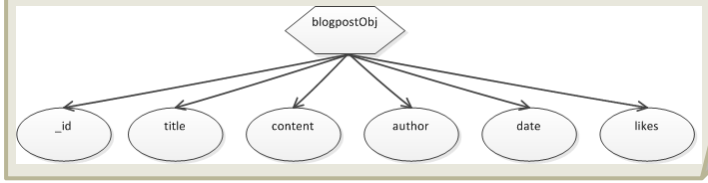
State-of-the-Art

	NoSQL-Datenbankmanagementsysteme						Schema-Bibliotheken	
	HBase	Cassandra	Datastore	MongoDB	CouchDB	Couchbase	KijiSchema für HBase	Python DM Lib für Datastore
Entity-Typen	(✓)	✓	✓	–	–	–	✓	✓
Constraints								
existence constraint	–	–	–	–	–	–	–	✓
unique constraint	–	–	–	–	–	–	–	–
primary key constraint	–	✓	–	–	–	–	✓	✓
domain constraints	(✓)	✓	–	–	–	–	✓	✓
ref. integrity constraints	–	–	–	–	–	–	–	✓
Validierung	(✓)	✓	–	–	–	–	✓	✓
Schema-Evolution								
add entity type	(✓)	✓	(✓)	–	–	–	✓	✓
delete entity type	(✓)	✓	–	–	–	–	✓	–
rename entity type	–	–	–	–	–	–	–	–
update entity type								
add attribute	–	✓	–	–	–	–	✓	✓
delete attribute	–	✓	–	–	–	–	✓	–
rename attribute	–	–	–	–	–	–	✓	–
move attribute	–	–	–	–	–	–	–	–
copy attribute	–	–	–	–	–	–	–	–
Mehrere Schemaversionen	–	–	–	–	–	–	✓	–
Datenmigration	–	–	–	–	–	–	–	–
Schema-Extraktion	–	–	–	–	–	–	–	–

Aus: Meike Klettke, Stefanie Scherzinger, Uta Störl: Datenbanken ohne Schema? - Herausforderungen und Lösungs-Strategien in der agilen Anwendungsentwicklung mit schema-flexiblen NoSQL-Datenbanksystemen. Datenbank-Spektrum 14(2): 119-129, 2014

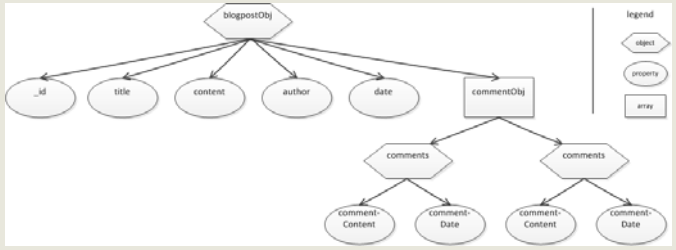
JSON-Beispiel (für heterogene Dokumente)


```
{"_id": 7,
"title": "NoSQL Data Modeling
Techniques",
"content": "NoSQL databases ...",
"author": "Alice",
"date": "2014-01-22",
"likes": 34 }
```

Version 1:
einfach strukturierte Blogposts


```
{"_id": 97,
"title": "NoSQL Schema Design",
"content": "Schema-flexibility ...",
"author": "Bob",
"date": "2014-02-24",
"comments": [
  { "commentContent": "Not sure...",
  "commentDate": "2014-02-24" },
  { "commentContent": "Let me ...",
  "commentDate": "2014-02-26" } ] }
```

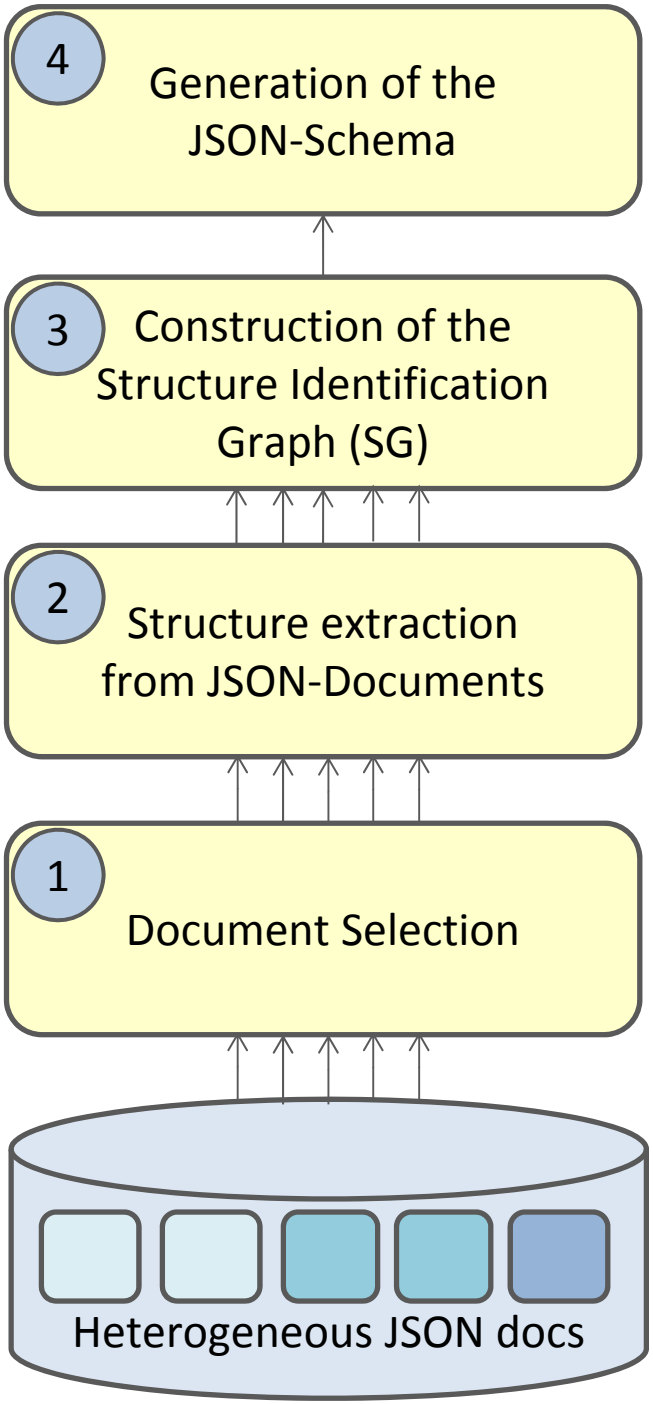



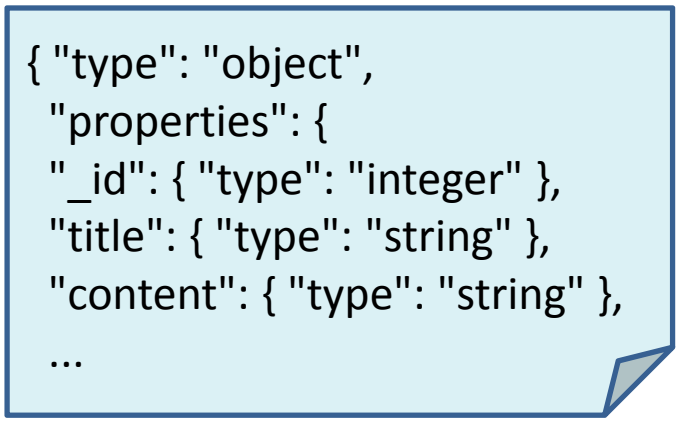
Version 2:
Blogposts mit Kommentaren

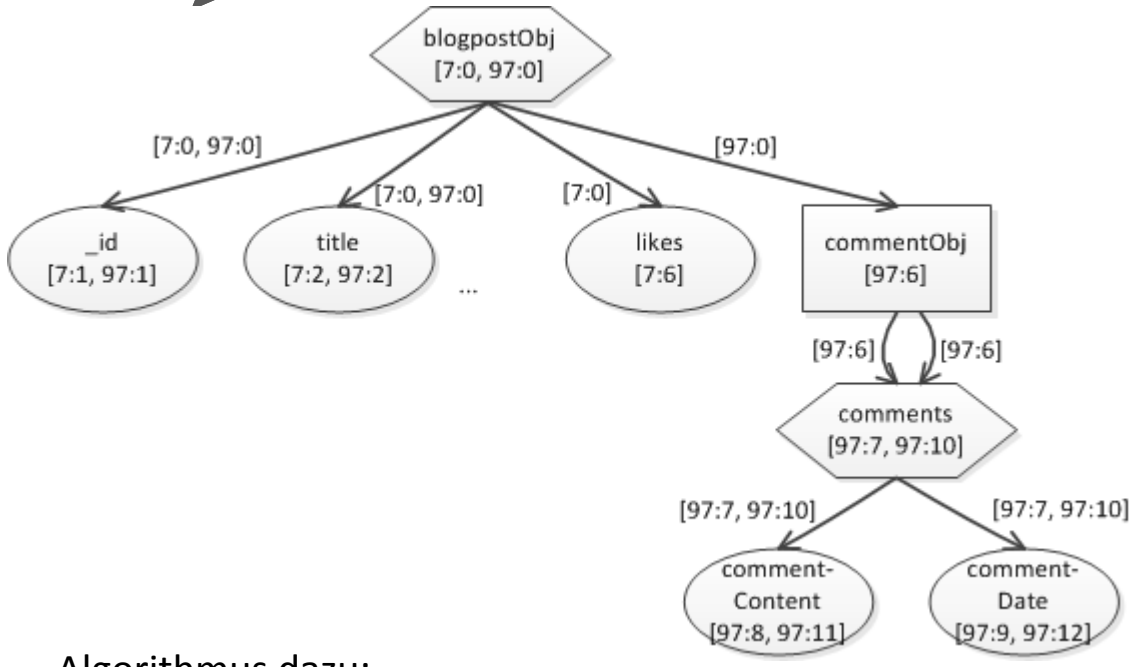
Ziel: Schemaableitung

- Verwendung von **JSON-Schema**
- Repräsentation von allen in der NoSQL-Datenbank enthaltenen Varianten

```
{ "type": "object",
  "properties": {
  "_id": { "type": "integer" },
  "title": { "type": "string" },
  "content": { "type": "string" },
  ...
  "comments": { "type": "array",
    "items": { "type": "object",
    "properties": {
    "commentContent": { "type": "string" },
    "commentDate": { "type": "string" } }
    "required": ["commentContent",
"commentDate"] } } }
  "required": ["title", "content", "author",
"date"] }
```

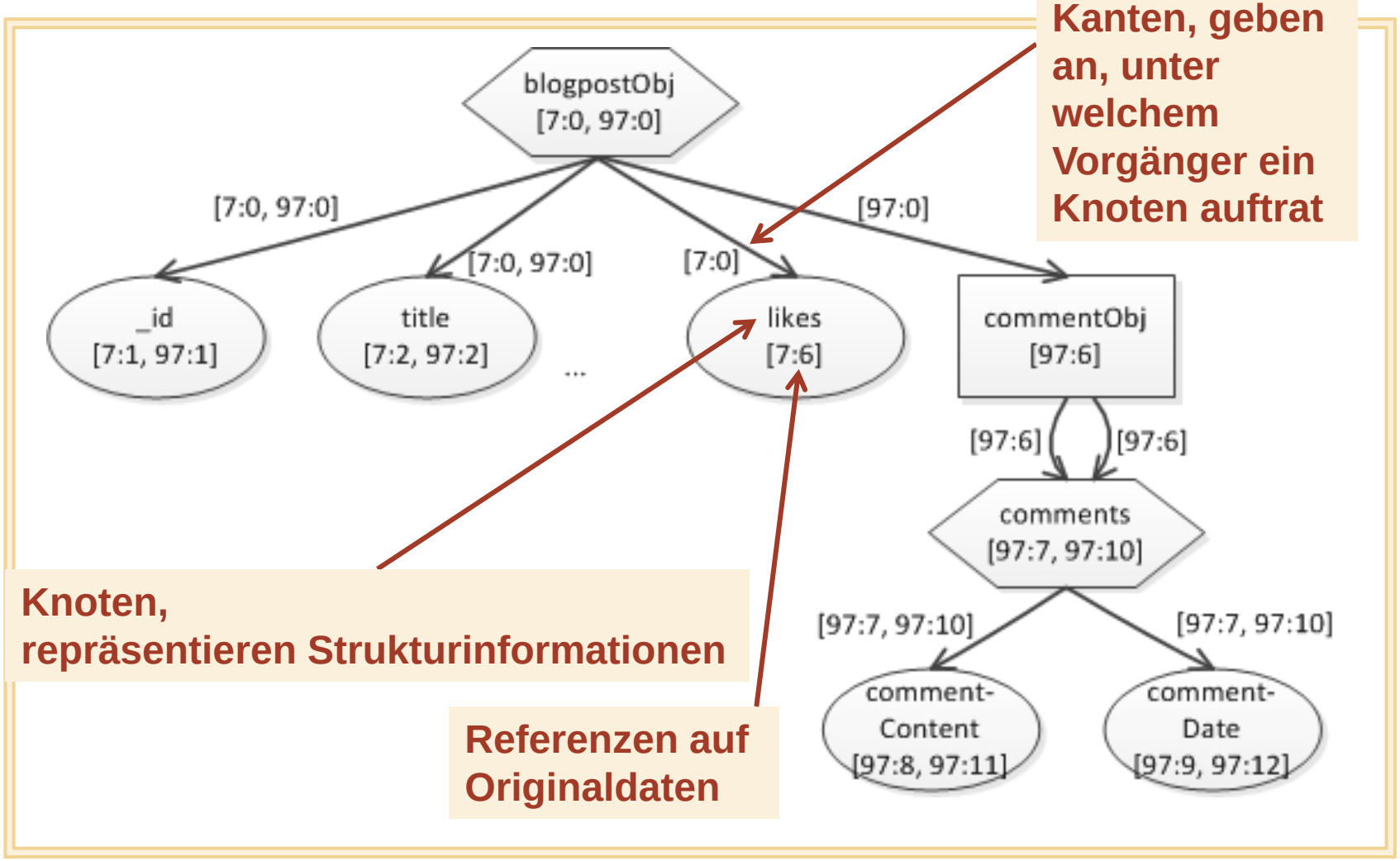






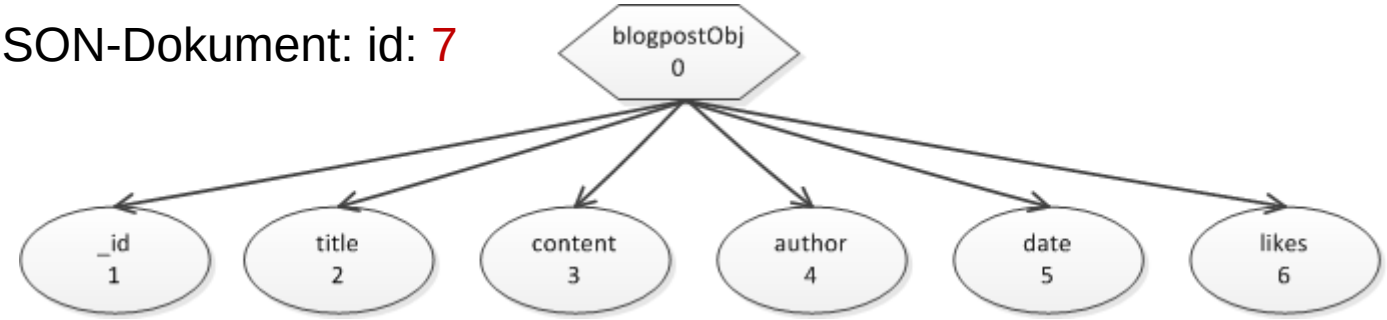
Algorithmus dazu:
Inspiriert von **XML** (**DTD-Ableitung**):
Chuang-Hue Moh, Ee-Peng Lim, Wee-Keon Ng: DTD-Miner, A Tool for Mining DTD from XML Documents. In Proc. WECWIS, 2000

I) Structure Identification Graph (SG)

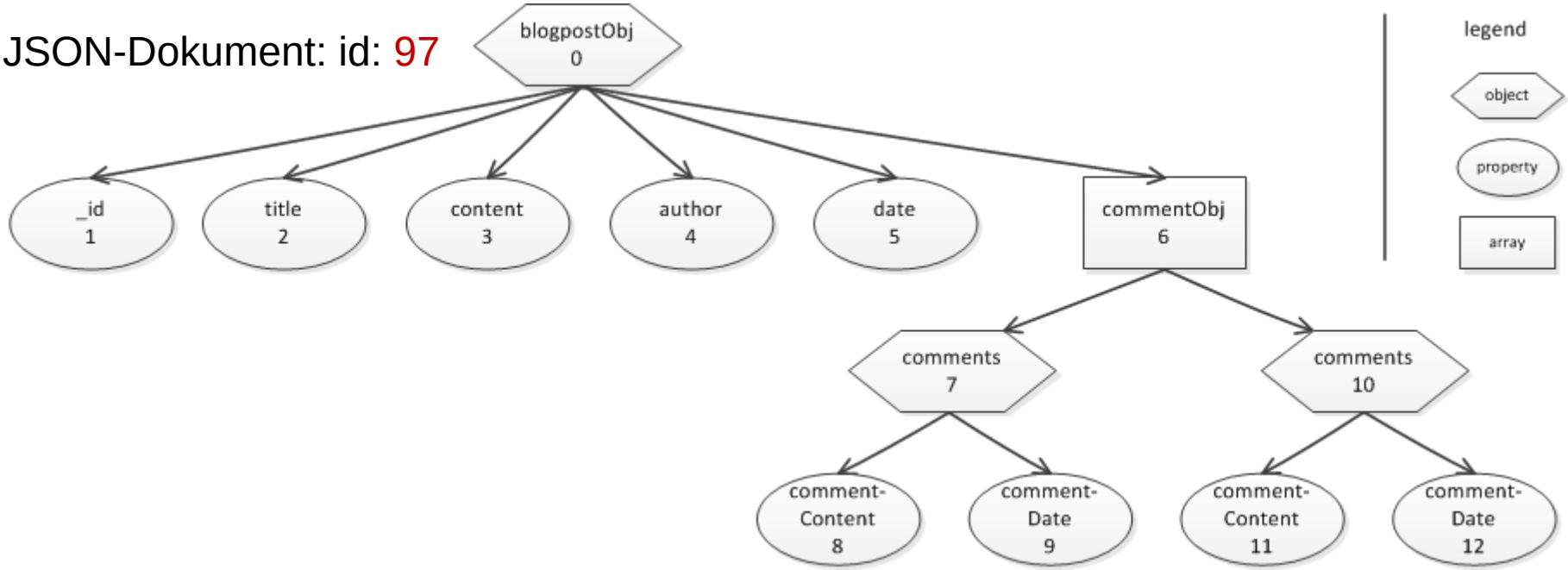


Knotennummerierung

JSON-Dokument: id: 7



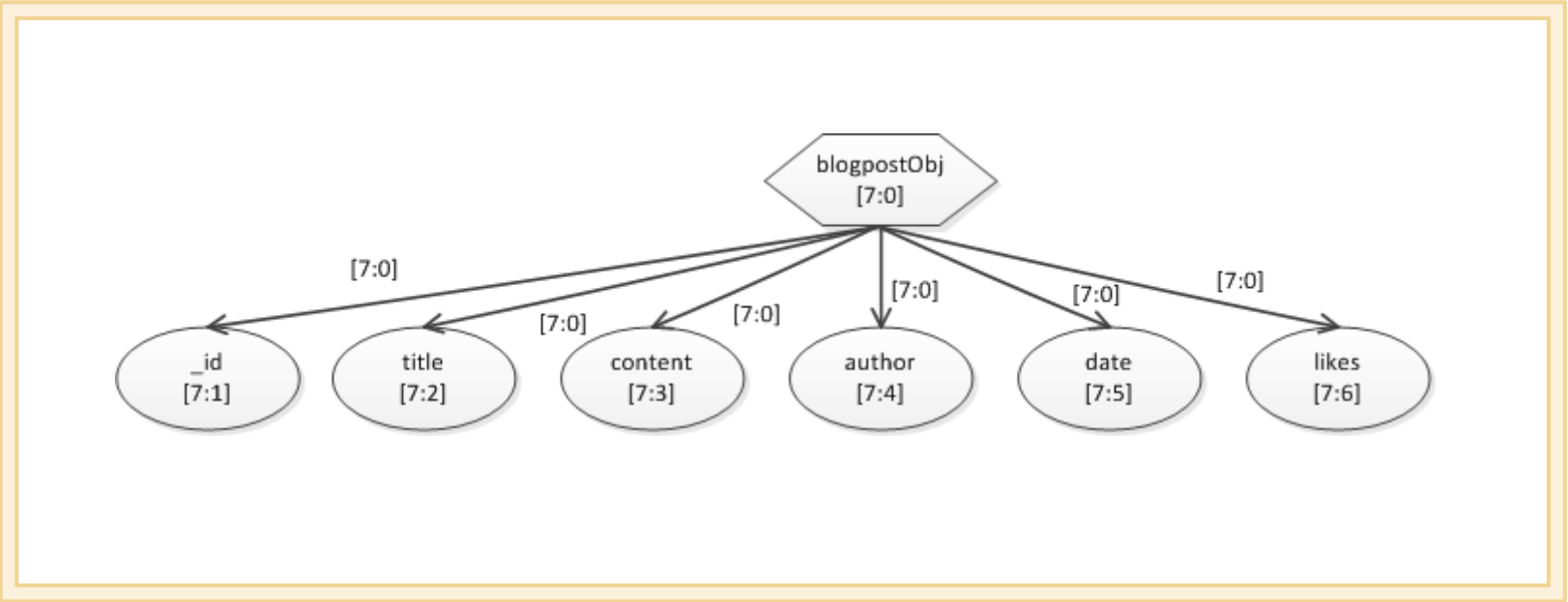
JSON-Dokument: id: 97



Aufbau des SG (1/3)

{"_id": 7,
"title": "NoSQL Data Modeling Techniques",
"content": "NoSQL databases ...",
"author": "Alice",
"date": "2014-01-22",
"likes": 34 }

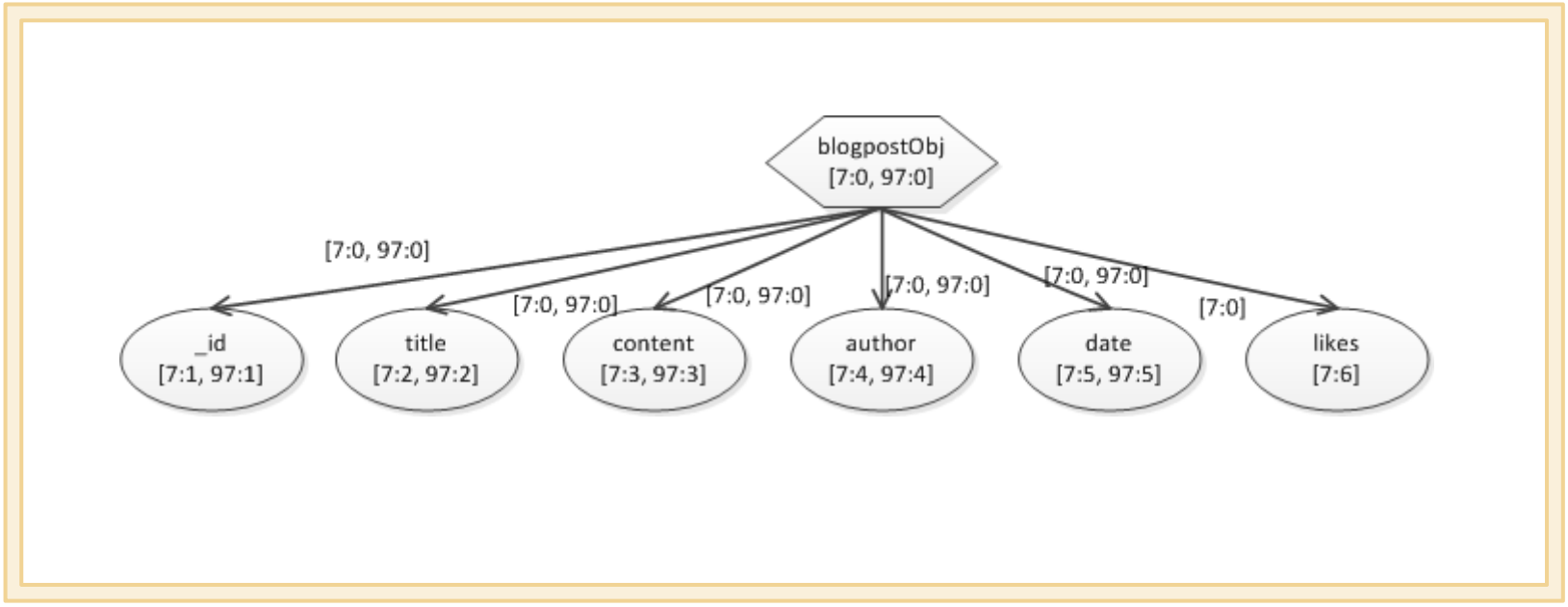
JSON-Dokument: id: 7



Aufbau des SG (2/3)

JSON-Dokument: id: 97

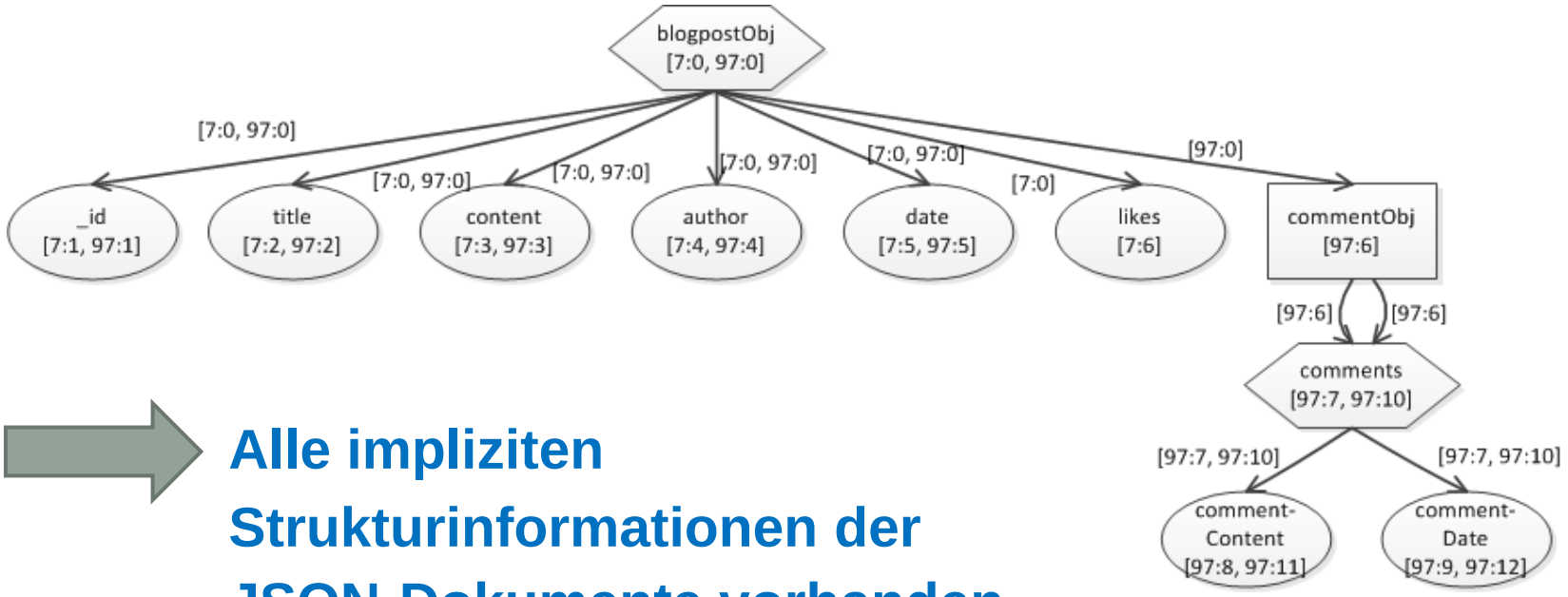
```
{"_id": 97,
"kind": "BlogPost",
"title": "NoSQL Schema Design",
"content": "Schema-flexibility ...",
"author": "Bob",
"date": "2014-02-24",
…}
```



Aufbau des SG (3/3)

```
…
"comments": [
  { "commentContent": "Not sure...",
  "commentDate": "2014-02-24" },
  { "commentContent": "Let me ...",
  "commentDate": "2014-02-26" } ] }
```

JSON-Dokument: id: 97

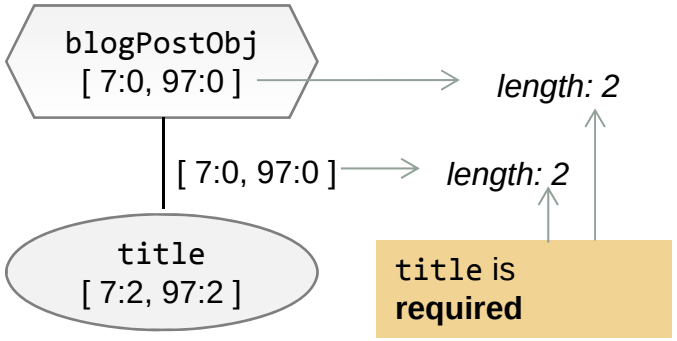


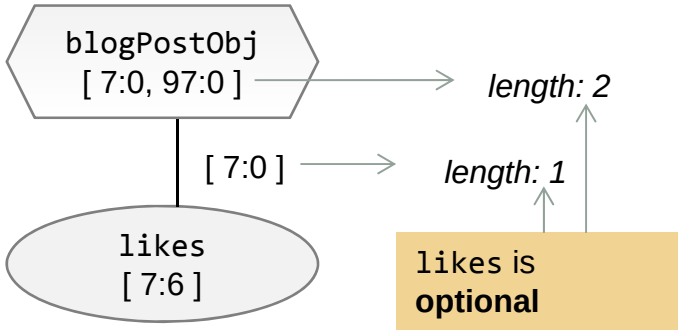
Alle impliziten Strukturinformationen der JSON-Dokumente vorhanden

II) Ableitung des JSON Schemas aus dem SG

- Ableiten der Struktur,
- Bestimmen der Properties, die required und optional sind aus den Knoten- und Kantenlisten

Beispiele:





- Und dann noch: Typableitung

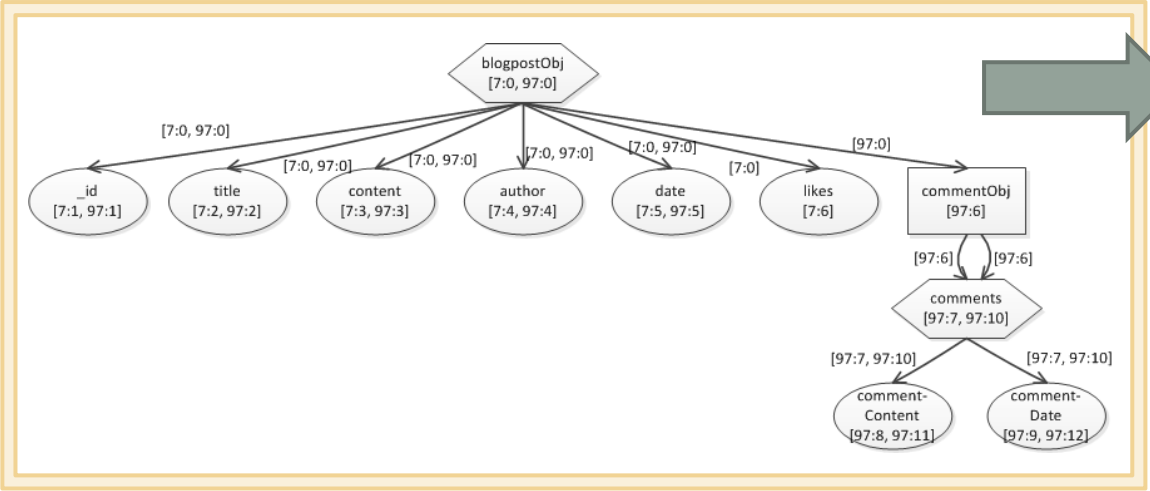
III) „Wer als Werkzeug nur einen Hammer hat, sieht in jedem Problem einen Nagel.“

Paul Watzlawick
Mark Twain

- Schemaextraktion ist vielfältig einsetzbar:
  - Auf **vollständigen Kollektionen**
  - Auf einem **Ausschnitt der Daten**
    - selektiert nach einer bestimmten **Merkmalskombination**
    - ausgewählt nach einer **Versionsnummer** oder
    - eingeschränkt auf einen **Zeitbereich** der Speicherung (timestamp)
- Keine Änderungen des Schemaextraktionsverfahrens erforderlich

Verwendung des SG zur Ausreißererkennung

- Ausreißer: Sind ebenfalls aus dem Structure Identification Graph (SG) ableitbar

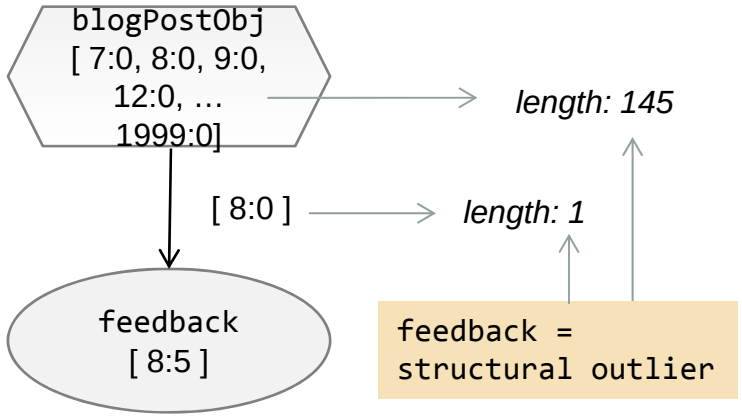


- Sehr selten auftretende keys
- Keys, die nur in sehr wenigen Fällen fehlen
- Keys, die nur in alten Versionen vorhanden sind

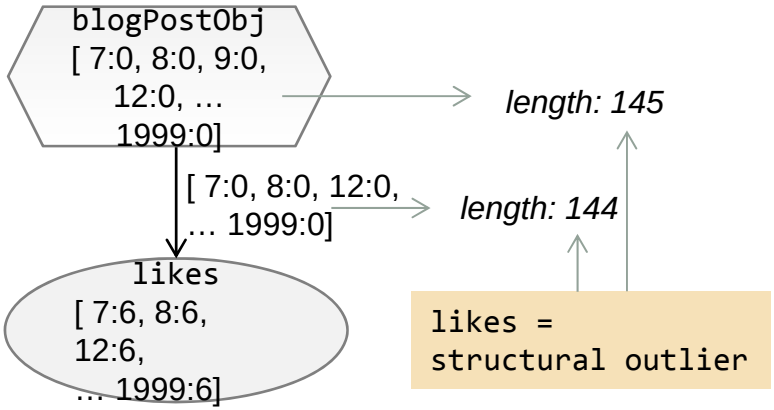
- Zu beachten: Ausreißer und „seltene Muster“ sind nicht zu unterscheiden

Ausreißererkennung: Variety vs. Veracity

Selten auftretende Keys



Selten fehlende Keys



$$\frac{|edgeList|}{|parentList|} * 100 < \epsilon$$

$$(\frac{|edgeList|}{|parentList|} * 100 > 100 - \epsilon)$$

Verfahren zur Erkennung:

Traversierung durch den SG

Für jeden Knoten: Vergleich

- Länge der Kantenliste (Kante zwischen Knoten und Parentknoten) mit
- Länge der Liste am Parentknoten

Ausreißererkennung

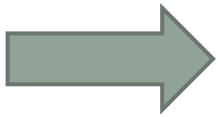
Nicht automatisch möglich, da keine Unterscheidung zwischen:

- Fehlern in den Daten und
- Selten vorkommenden Strukturen

möglich

Semiautomatischer Prozess:

1. Finden von seltenen Mustern (Kandidaten) = unter einem gegebenen Grenzwert ε
2. Dialog mit dem Benutzer, ob diese korrekt sind oder nicht

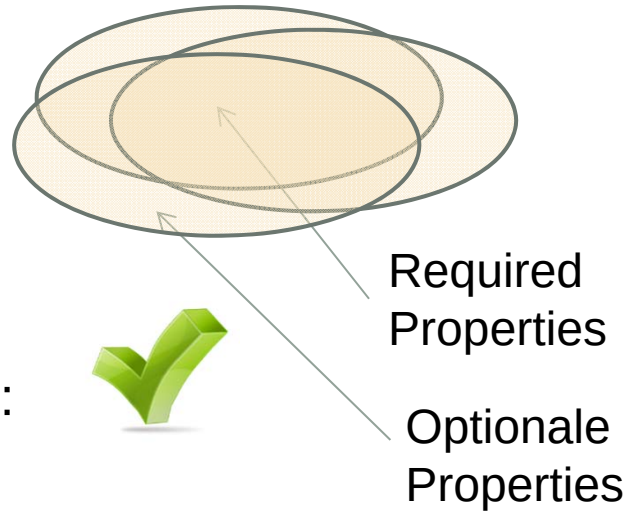


Fehlerklassifikation erfolgt also letztendlich durch den Anwender

Verwendung des SG zur Ableitung von Measures

- Ziel: Bestimmung des **Grades der Strukturierung**: wie regulär oder wie heterogen sind die Daten?

$$cov(J_1..J_k) = \frac{1}{k} \cdot \sum_{i=1}^{k} \frac{|\bigcap_{j=1}^{k} J_j|}{|J_i|}$$

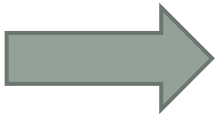


$\bigcap_{j=1}^{n} J_j$ = Anzahl der Required properties:

$|J_i|$ = lässt sich aus dem Spanning Graph ableiten, aber noch einfacher beim Parsen der JSON-Dokumente speichern

Bewertung des Verfahrens

- Effektives Verfahren zur Schemaableitung
- Nachteil:
  - Große interne Graphstruktur
  - bei großen Eingabe-Collections sehr lange Listen an Knoten- und Kantenlisten

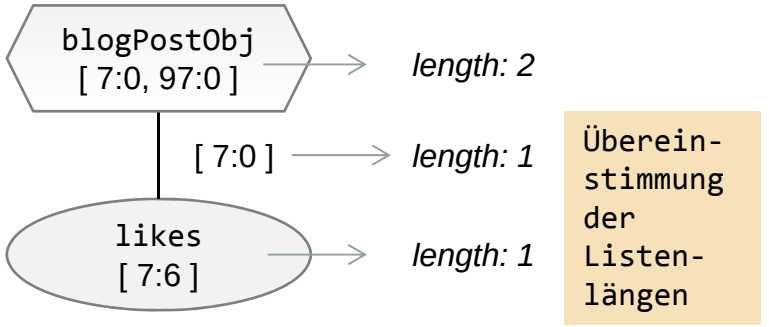


Vereinfachung notwendig

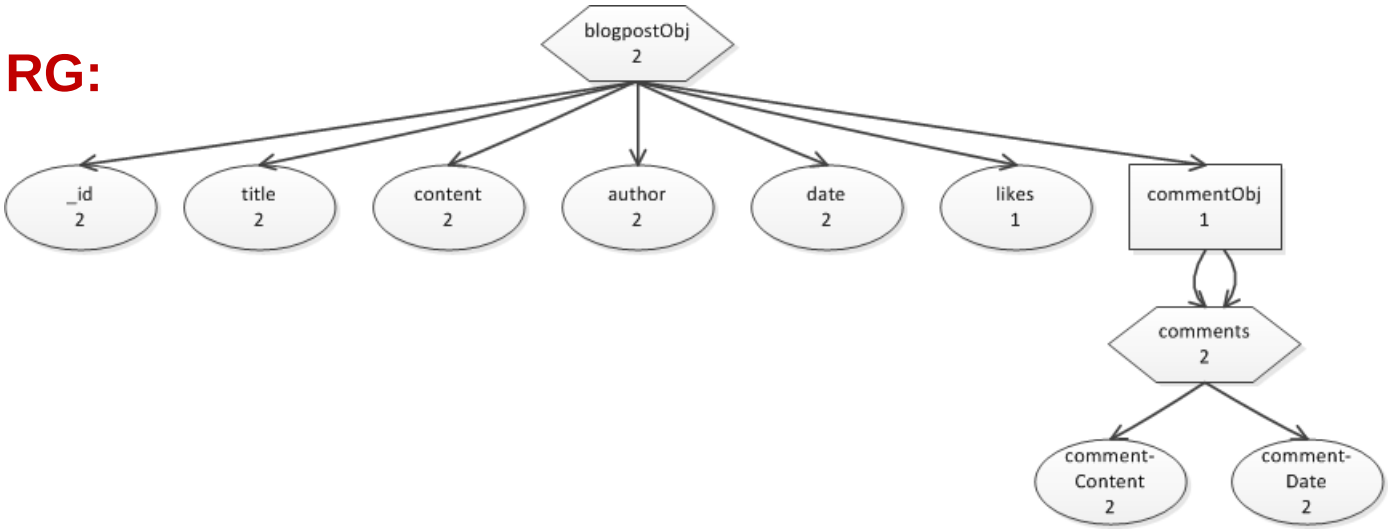
Reduced Structure Identification Graph (RG)

IV) Reduced Structure Identification Graph (RG)

- Knoten- und Kantenlabels: keine Listen mit Referenzen auf die Originaldokumente, nur die Häufigkeit des Auftretens
- Da diese bei jedem Knoten mit der Listenlänge der Kante (vom Knoten zum Parentknoten) übereinstimmt, sind Kantenlabels nicht notwendig



- **Beispiel für RG:**



Verwendung von SG und RG

	SG – Structure Identifikation Graph	RG – Reduced Structure Identifikation Graph
JSON Schema Extraktion	+	+
Dokument Statistiken	+	+
Ähnlichkeitsmaße (Coverage)	+	+
Ausreißererkennung	+	-

Ableitbar durch einen zweistufigen Prozess:

1. Dokument-Statistiken ableiten
2. Ausreißer durch gezielte Anfragen finden: (z.B. in MongoDB)

```
db.collection.find({pi: {$exists: true}}
db.collection.find({pj: {$exists: false}})
```

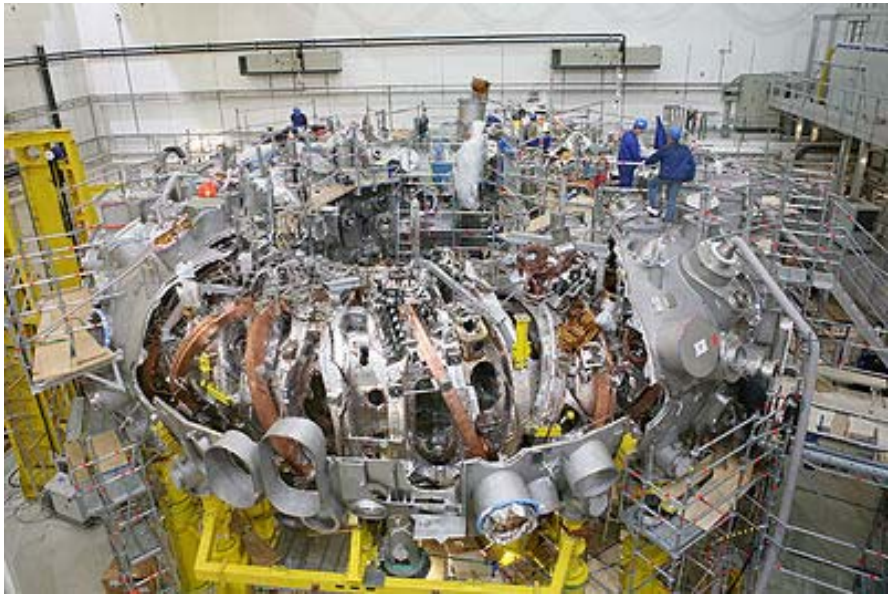
V) Implementierung und Performance-Messungen

- Python, pymongo, MongoDB
- Daten von IPP Greifswald

  (Experimentdaten vom September 2014)
  - MongoDB
  - 550MB Daten
  - In 180 Collections
- Hardware:

  Intel Core i7-461000U CPU @ 2.70GHz

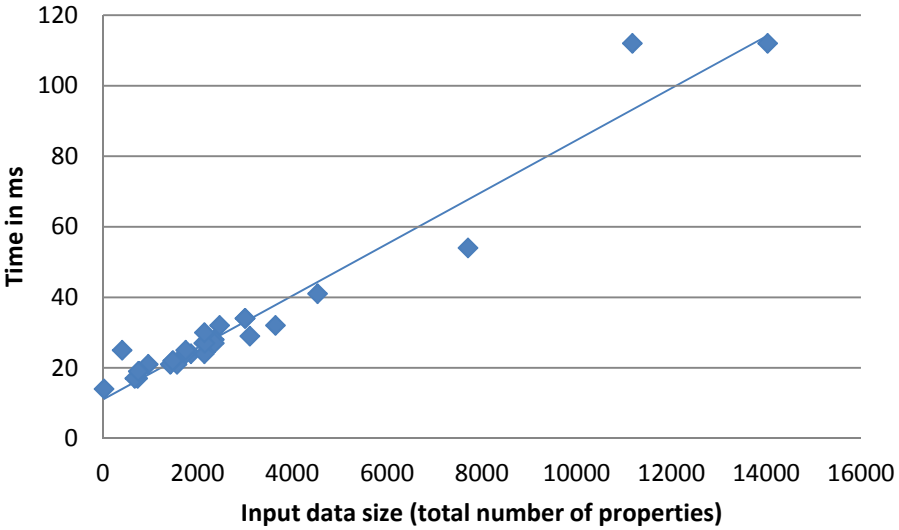
  8.00 GB RAM



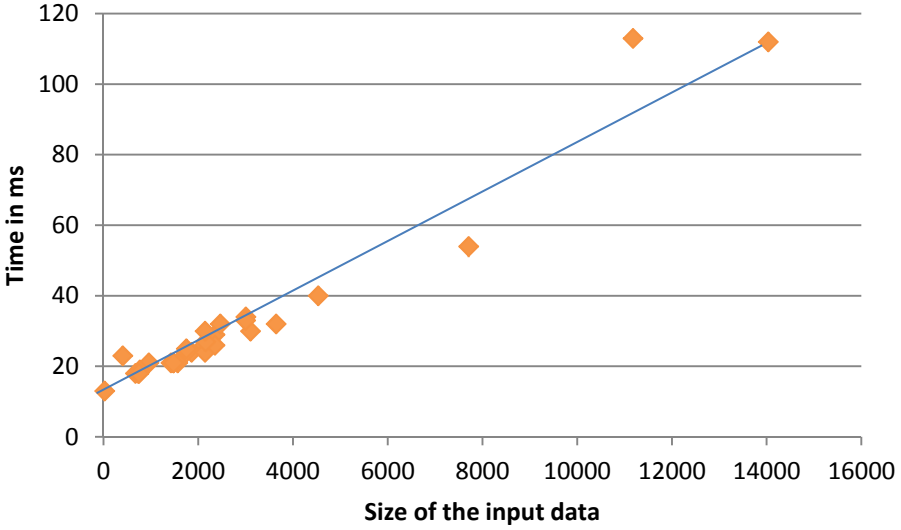
- Schemaableitung, Aufbau des SG und RG für verschiedene Dokumentkollektionen

Ergebnisse

- Für die 30 „datenintensivsten“ Collections aus den Experimentdaten (IPP Greifswald)



Aufbau des SG

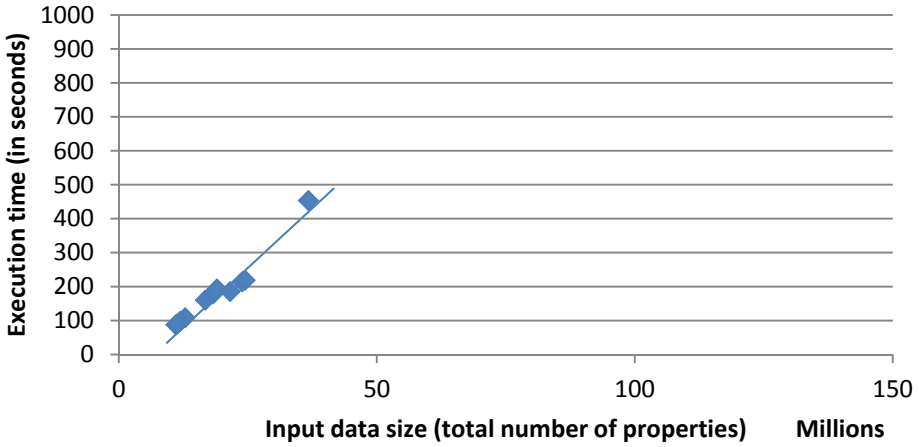


Aufbau des RG

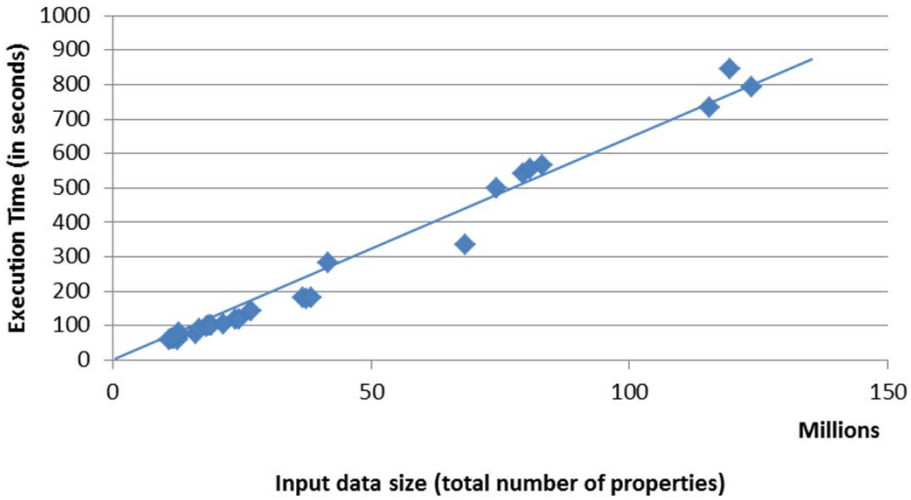
Keine messbaren Unterschiede

Mehr Ergebnisse

- Für eine größere Datenbank (Web Performance Data)



Aufbau des SG



Aufbau des RG

messbaren Unterschiede bei großen Eingangsdaten

SG nur für Eingangsdaten bis
ca. 40 Mio. Eingangsproperties/ 500 Mbyte Daten

Zusammenfassung und Ausblick

Schemainformationen sind notwendig

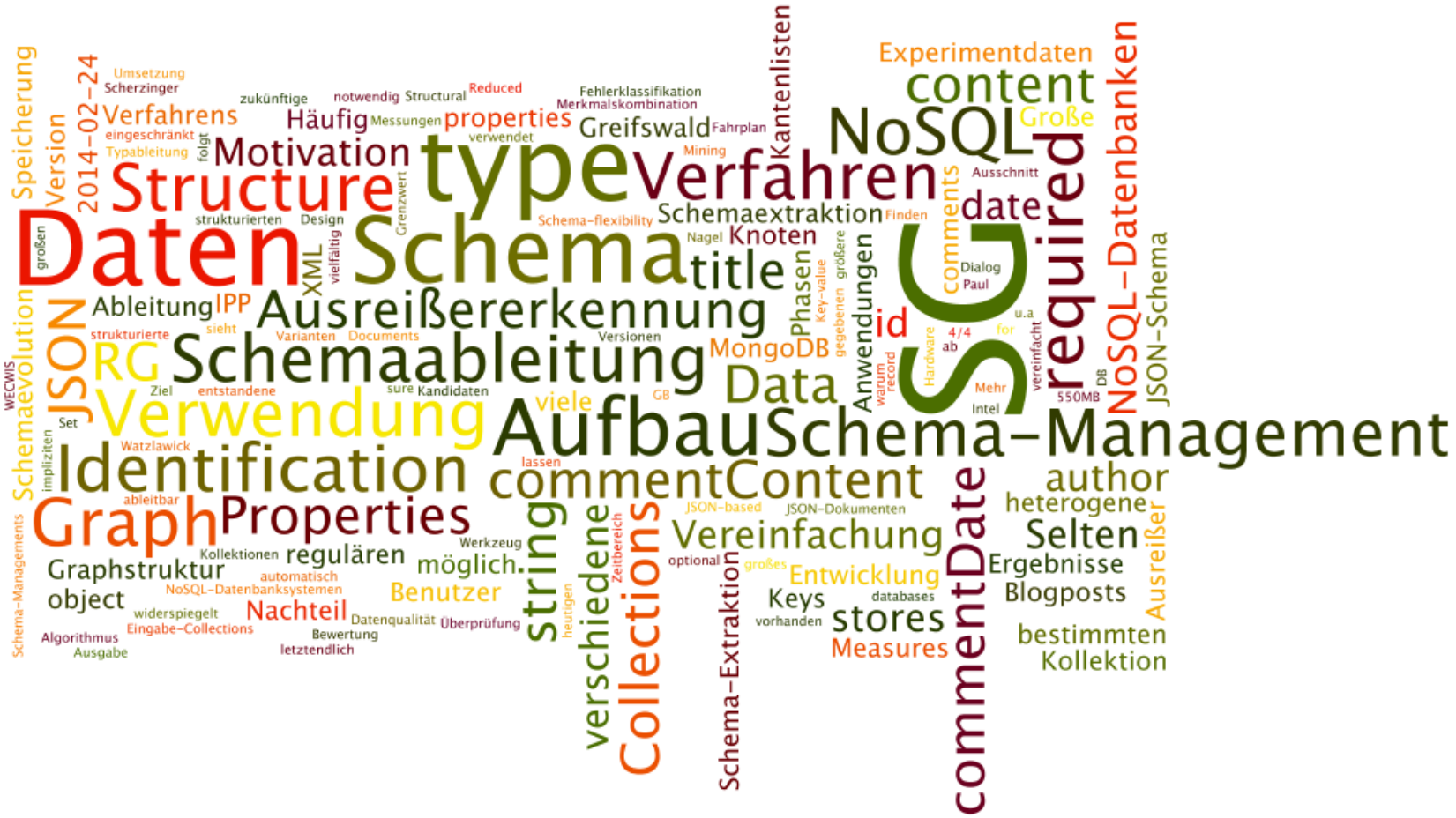
- damit **Anwendungen** darauf reagieren können
- zur Datenintegration (**in andere DB)**
- zur Überprüfung der **Datenqualität**
- **Zur Vereinfachung der Schemaevolution und Dokumentmigration**

Uta Störl et al.
BTW- Industrieprogramm

Stefanie Scherzinger, et al.
Cleager: **Eager Schema Evolution in NoSQL Data Stores, BTW**

Vielen Dank fürs Zuhören !
Zeit für Fragen …

….. zu diesen und anderen Themen?



Weiterführende Literatur

- C.-H. Moh, E.-P. Lim, and W. K. Ng. DTD-Miner: A Tool for Mining DTD from XML Documents. In WECWIS, pages 144{151, 2000.
- JSON Schema, 2013. http://json-schema.org/.
- Stefanie Scherzinger, Meike Klettke, Uta Störl: *Cleager: Eager Schema Evolution in NoSQL Document Stores*, Demoprogramm, BTW 2015
- Uta Störl, Thomas Hauf, Meike Klettke, Stefanie Scherzinger: *Schemaless NoSQL Data Stores – Object-NoSQL Mappers the Rescue?* Industrieprogramm, BTW 2015
- Meike Klettke, Stefanie Scherzinger, Uta Störl*: Datenbanken ohne Schema? - Herausforderungen und Lösungs-Strategien in der agilen Anwendungsentwicklung mit schema-flexiblen NoSQL-Datenbanksystemen.* Datenbank-Spektrum 14(2): 119-129
- Stefanie Scherzinger, Meike Klettke, and Uta Störl*: Managing Schema Evolution in NoSQL Data Store,* The 14th International Symposium on Database Programming Languages DBPL@VLDB, Italy, 2013
- Meike Klettke: Modellierung, *Bewertung und Evolution von XML-Dokumentkollektionen.* Habilitationsschrift, Universität Rostock, Fakultät für Informatik und Elektrotechnik, 2007

Literatur

- Apache Cassandra, 2013. http://cassandra.apache.org/.
- V. Benzaken, G. Castagna, K. Nguyen, and J. Simeon. “Static and dynamic semantics of NoSQL languages”. In Proc. POPL, pages 101–114, 2013.
- M. Brown. Developing with Couchbase Server. O’Reilly, 2013.
- R. Cattell. “Scalable SQL and NoSQL data stores”. SIGMOD Record, 39(4):12–27, 2010.
- F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, et al. “Bigtable: A Distributed Storage System for Structured Data”. In Proc. OSDI, pages 205–218, 2006.
- K. Chodorow. MongoDB: The Definitive Guide. O’Reilly, 2013.
- Couch Potato, 2010. https://github.com/langalex/couch_potato/issues/14.
- Google Inc. Google Datastore Admin, 2013. https://developers.google.com/appengine/docs/adminconsole/datastoreconsole.

Literatur

- Google Inc. Google Datastore, 2013. https://developers.google.com/appengine/docs/java/datastore/.
- Google Inc. Using JDO with App Engine, 2013. https://developers.google.com/appengine/docs/java/datastore/jdo/.
- D. Habich, M. Boehm, M. Thiele, B. Schlegel, U. Fischer, H. Voigt, and W. Lehner. “Next Generation Database Programming and Execution Environment”. In Proc. DBPL, 2011.
- JBoss Community. Hibernate OGM (Object/Grid Mapper), 2013. http://www.hibernate.org/subprojects/ogm.html.
- JSON.org. Introducing JSON, 2013. http://www.json.org/.
- S. Lightstone. Making it Big in Software. Prentice Hall, 2010.
- J. McWilliams and M. Ivey. Google Developers: Updating your model’s schema, 2012. https://developers.google.com/appengine/articles/update_schema.
- Morphia. A type-safe java library for MongoDB, 2013. http://code.google.com/p/morphia/.

Literatur

- Objectify AppEngine. Migrating Schemas, 2012. https://code.google.com/p/objectify-appengine/wiki/SchemaMigration.
- Objectify AppEngine. The simplest convenient interface to the Google App Engine datastore, 2013. https://code.google.com/p/objectify-appengine/.
- D. L. Parnas. “Software Aging”. In Proc. ICSE, pages 279–287, 1994.
- E. Redmond and J. R. Wilson. Seven Databases in Seven Weeks: A Guide to Modern Databases and the NoSQL Movement. Pragmatic Bookshelf, 2012.
- J. Robie, D. Chamberlin, M. Dyck, D. Florescu, J. Melton, and J. Simeon. XQuery Update Facility 1.0, 2011. http://www.w3.org/TR/xquery-update-10/.
- J. Robie, G. Fourny, M. Brantner, D. Florescu, T. Westmann, and M. Zaharioudakis. JSONiq Grammar - Extensions, 2013. http://www.jsoniq.org/grammars/extension.xhtml.
- Stack Overflow, 2012. http://stackoverflow.com/questions/8920610/add-a-new-attribute-to-entity-in-datastore.
- S. Tiwari. Professional NoSQL. John Wiley & Sons, 2011.

V Weitere und zukünftige Arbeiten

- Hybride Speicherung in SQL und NoSQL-Datenbanken
  - Auswahl des geeigneten Datenbanksystems für komplexe Anwendungen
  - Aufteilen des Datenbestandes in verschiedene Systeme
- Kombination von Schemaextraktion und Validierung mit Object-to-NoSQL-Mappern
- Offene Aufgabe für Datenbanksysteme:

  Objekt-relational Gap