

Big Data: Neue Entwicklungen im Datenbankbereich

- **Semantic Web: RDF**
- **Datenströme**
- **Information Retrieval**
- **Graph Exploration**
- **Map Reduce: Massiv parallele Verarbeitung**
- **Peer to Peer Informationssysteme**
- **Key Value Stores / NoSQL**
- **Multi-Tenancy/Cloud-Datenbanken**

1

21 Big Data	737
21.1 Datenbanken für das Semantic Web	737
21.1.1 RDF: Resource Description Framework	737
21.1.2 SPARQL: Die RDF Anfragesprache	740
21.1.3 Implementierung einer RDF-Datenbank	742
21.2 Datenströme	746
21.3 Information Retrieval und Suchmaschinen	751
21.3.1 TF-IDF: Dokument-Ranking basierend auf Begriffs-Häufigkeit	752
21.3.2 Invertierte Indexierung	754
21.3.3 Page Rank	754

16

Inhaltsverzeichnis

21.3.4 Der HITS Algorithmus	757
21.4 Graph-Exploration (Graph Mining)	760
21.4.1 Darstellung von Graphen	760
21.4.2 Zentralitätsmaße	763
21.4.3 Verbindungs-Zentralität (Degree Centrality)	763
21.4.4 Nähe-Zentralität (Closeness Centrality)	764
21.4.5 Pfad-Zentralität (Betweenness Centrality)	765
21.5 MapReduce: Massiv parallele Datenverarbeitung	766
21.6 Peer-to-Peer-Informationssysteme	770
21.6.1 P2P-Systeme für den Datenaustausch (File-Sharing)	771
21.6.2 Verteilte Hashtabellen (Distributed Hash Tables DHTs)	773
21.6.3 Mehrdimensionaler P2P-Datenraum	777
21.7 No-SQL- und Key/Value-Datenbanksysteme	778
21.8 Multi-Tenancy, Cloud Computing und Software as a Service	780

2

Semantic Web: Resource Description Framework (RDF)

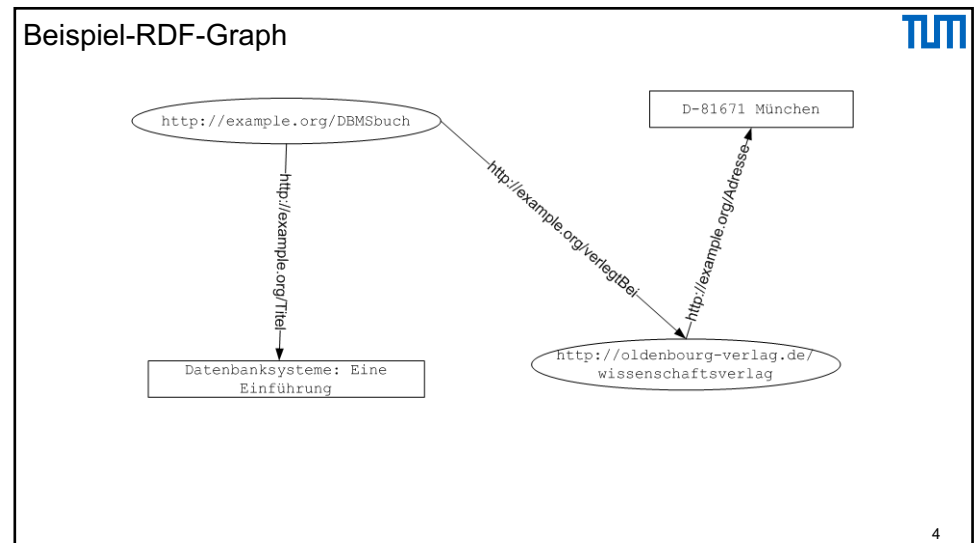
Triple-Datenmodell

- (Subjekt, Prädikat, Objekt)

Meist graphische Visualisierung

- Subjekte und Objekte sind Knoten
- Prädikate sind gerichtete Kanten
 - Von Subjekt-Knoten nach Objekt Knoten

3





- Das Buch mit der Kennung `<http://example.org/DBMSbuch>` wird `<http://example.org/verlegtBei>` dem Verlag `<http://oldenbourg-verlag.de/wissenschaftsverlag>`.
- Der `<http://oldenbourg-verlag.de/wissenschaftsverlag>` hat als `<http://example.org/Adresse>` das Literal "D-81671 München".
- Das `<http://example.org/DBMSbuch>` hat als `<http://example.org/Titel>` das Literal "Datenbanksysteme: Eine Einführung".

Somit wird klar, dass das erste Element des Tripels das Subjekt einer Aussage, das zweite das Prädikat und das dritte Element das Objekt darstellt.
 Es gibt mehrere leicht unterschiedliche, textuelle RDF-Tripeldarstellungen, die man unter den Namen *N3*, *N-Triples* oder *Turtle* findet. In unserem Fall sieht die Beispieldatenbank in der *Turtle*-Notation wie folgt aus:

```
<http://example.org/DBMSbuch> <http://example.org/verlegtBei>
  <http://oldenbourg-verlag.de/wissenschaftsverlag>.
<http://oldenbourg-verlag.de/wissenschaftsverlag>
  <http://example.org/Adresse> "D-81671 München".
<http://example.org/DBMSbuch> <http://example.org/Titel>
  "Datenbanksysteme: Eine Einführung".
```

5



Kurzform

```
@prefix ex: <http://example.org>.
@prefix ol: <http://oldenbourg-verlag.de>.
```

```
ex:DBMSbuch ex:verlegtBei ol:wissenschaftsverlag.
ol:wissenschaftsverlag ex:Adresse "D-81671 München".
ex:DBMSbuch ex:Titel "Datenbanksysteme: Eine Einführung".
```



```
ex:DBMSbuch ex:verlegtBei ol:wissenschaftsverlag;
  ex:Titel "Datenbanksysteme: Eine Einführung".
ol:wissenschaftsverlag ex:Adresse "D-81671 München".
```

Man beachte, dass man jetzt ein Semikolon verwendet, um anzugeben, dass das vorherige Subjekt übernommen wird. Man kann mengenwertige Beziehungen auch noch weiter kompaktifizieren, wenn man Cluster gleicher Subjekte und Prädikate bildet. Beispielsweise:

```
ex:DBMSbuch ex:AutorNachName "Kemper" ,
  "Eickler".
```

6



Namenlose Knoten

```
@prefix ex: <http://example.org>.
@prefix xsd: <http://www.w3.org/2001/XMLSchema#>.
```

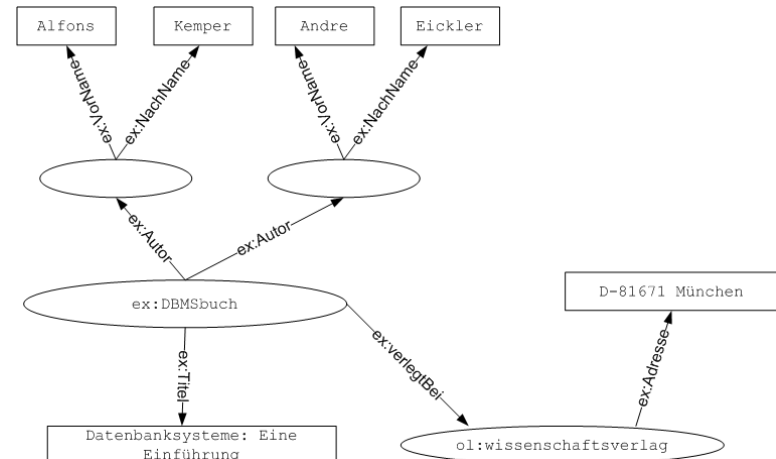
```
ex:DBMSbuch ex:Autor _:k.
_:k ex:NachName "Kemper"^^xsd:string.
_:k ex:VorName "Alfons"^^xsd:string.
```

```
ex:DBMSbuch ex:Autor _:e.
_:e ex:NachName "Eickler"^^xsd:string.
_:e ex:VorName "Andre"^^xsd:string.
```

7



Graph mit unbenannten Knoten



8

SPARQL: Die RDF Anfragesprache



```
SELECT ?Var1 ?Var2 ... $VarN
WHERE {Muster1. Muster2. ... MusterM. }
```

```
PREFIX ex: <http://www.example.org>
```

```
SELECT ?AutorenDesOldenbourgVerlags WHERE
{ ?buch ex:Autor ?a.
  ?a ex:NachName ?AutorenDesOldenbourgVerlags.
  ?buch ex:verlegtBei <http://oldenbourg-verlag.de/wissenschaftsverlag>.
}
```

9

SPARQL: Die RDF Anfragesprache



```
PREFIX ex: <http://www.example.org>
```

```
SELECT ?KempersBuecherTitel WHERE
{ ?KempersBuecher ex:Autor ?k.
  ?k ex:NachName "Kemper".
  ?KempersBuecher ex:Titel ?KempersBuecherTitel.
}
```

10

SPARQL: Die RDF Anfragesprache



Union

```
PREFIX ex: <http://www.example.org>
```

```
SELECT ?KempersOderEicklersBuecherTitel WHERE
{ { ?KempersBuecher ex:Autor ?k.
  ?k ex:NachName "Kemper".
  ?KempersBuecher ex:Titel ?KempersOderEicklersBuecherTitel.
} UNION
{ ?EicklersBuecher ex:Autor ?k.
  ?k ex:NachName "Eickler".
  ?EicklersBuecher ex:Titel ?KempersOderEicklersBuecherTitel.
}}
```

11

SPARQL: Die RDF Anfragesprache



optional

```
PREFIX ex: <http://www.example.org>
```

```
SELECT ?KempersBuecherTitel ?KempersBuecherISBN WHERE
{ ?KempersBuecher ex:Autor ?k.
  ?k ex:NachName "Kemper".
  ?KempersBuecher ex:Titel ?KempersBuecherTitel.
  OPTIONAL { ?KempersBuecher ex:hatISBN ?KempersBuecherISBN }
}
```

12

SPARQL: Die RDF Anfragesprache



filter

```
PREFIX ex: <http://www.example.org>

SELECT ?KempersBuecherTitel ?auflagenNr WHERE
{
  ?KempersBuecher ex:Autor ?k.
  ?k ex:NachName "Kemper".
  ?KempersBuecher ex:Titel ?KempersBuecherTitel.
  ?KempersBuecher ex:Auflage ?auflagenNr.
  FILTER ( ?auflagenNr > 7 )
}
```

13

SPARQL: Die RDF Anfragesprache



count-Aggregation

PREFIX ex: <http://www.example.org>

```
SELECT COUNT ?verlag WHERE
{
  ?buch ex:verlegtBei ?verlag.
}
```

Für SQL-affine Leser ist diese Anfrage etwas gewöhnungsbedürftig, da man ja in der Tat die Bücher zählen will. Die SPARQL-Formulierung zielt aber darauf ab, die Anzahl der Vorkommnisse des Musters „**?buch ex:verlegtBei ?verlag**“ für jeden **?verlag** zu zählen.

14



15

Implementierung einer RDF-Datenbank: RDF-3X



Dictionary	
Literal/URI	Code
<http://www.example.org/DBMSbuch>	0
Datenbanksysteme: Eine Einführung	1
<http://www.example.org/Titel>	2
<http://www.example.org/verlegtBei>	3
<http://oldenbourg-verlag.de/wissenschaftsverlag>	4
<http://www.example.org/Adresse>	5
D-81671 München	6
<http://www.example.org/Autor>	7
<dummy1>	8
<dummy2>	9
<http://www.example.org/VorName>	10
<http://www.example.org/NachName>	11
Alfons	12
Kemper	13
Andre	14
Eickler	15

16

B-Bäume ... so viele wie möglich

OPS
(1,2,0) (4,3,0)(6,5,4)(8,7,0)(9,7,0) (12,10,8)(13,11,8)(14,10,9)(15,11,9)

SPO
(0,2,1) (-,3,4)(-,7,8)(-,9,0)(4,5,6)(8,10,12)(-,11,13)(9,10,14) (-,11,15)

POS
(2,1,0)(3,4,0)(5,6,4)(7,8,0)(-,9,0)(10,12,8)(-,14,9)(11,13,8)(-,15,9)

SO OS PO OP PS SP
(0,2,1)(-,3,1)(-,7,2)(4,5,1) ...

S P O
(0,4) ...

TUM

17

Kompressionstechnik: Dictionary und Präfix

Jedes Tripel (s,p,o) wird also genau 6 mal repliziert abgelegt -- allerdings in permutierter Subjekt/Prädikat/Objekt-Reihenfolge, nämlich

- (p,s,o), (s,p,o), (p,o,s), (o,s,p), (s,o,p) und (o,p,s).

Zusätzlich gibt es noch die sogenannten aggregierten Indexe, die die Anzahl der Vorkommen des jeweiligen Patterns repräsentieren.

Zum Beispiel bedeutet der Eintrag (s,o,7), dass das Subjekt s siebenmal mit dem Objekt o in einer Beziehung steht -- mit beliebigem Prädikat.

Das Speichervolumen wird dadurch (dramatisch) reduziert, dass man in den Blättern der Bäume eine Präfix-Komprimierung durchführt. Z.B. wird in dem zweiten Eintrag des SPO-Baums das Subjekt 0 weggelassen, da es identisch zum ersten Eintrag ist. In dem vierten Eintrag kann sogar die Subjekt- und die Prädikat-Kennung weggelassen werden, da beide identisch zum dritten Eintrag sind

TUM

18

Kompressionstechnik: Dictionary und Präfix

Es werden aber nicht nur gleiche Präfixe weggelassen; zusätzlich wird auch anstatt des jeweiligen Codes nur die Differenz zum Code des Vorgänger-Tripels gespeichert. Der letzte Eintrag im SPO-Baum würde demnach als (-,1,1) gespeichert, da er in der ersten Komponente identisch zum Vorgänger-Tripel ist, in der zweiten und dritten Komponente ist die Differenz zum Vorgänger-Tripel jeweils 1.

Diese Kompression ist sehr effektiv, da die Tripel in den Blattknoten ja fortlaufend sortiert sind und sich deshalb immer nur geringfügig vom Vorgänger-Tripel unterscheiden.

Als Anker für diese Differenz-Kompression wird auf jeder Blatt-Seite immer nur ein vollständiges Tripel, nämlich das Erste, gespeichert.

TUM

19

Anfrageauswertung

PREFIX ex: <http://www.example.org>

```
SELECT REDUCED ?AutorenDesOldenbourgVerlags WHERE
{ ?buch ex:Autor ?a.
  ?a ex:NachName ?AutorenDesOldenbourgVerlags.
  ?buch ex:verlegtBei <http://oldenbourg-verlag.de/wissenschaftsverlag>.
}
```

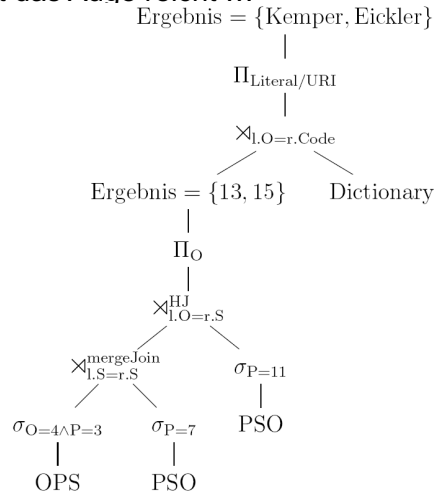
Dict-Lookup

```
SELECT REDUCED ?AutorenDesOldenbourgVerlags WHERE
{ ?buch 7 ?a.
  ?a 11 ?AutorenDesOldenbourgVerlags.
  ?buch 3 4.
}
```

TUM

20

Merge-Joins ... so weit das Auge reicht ...

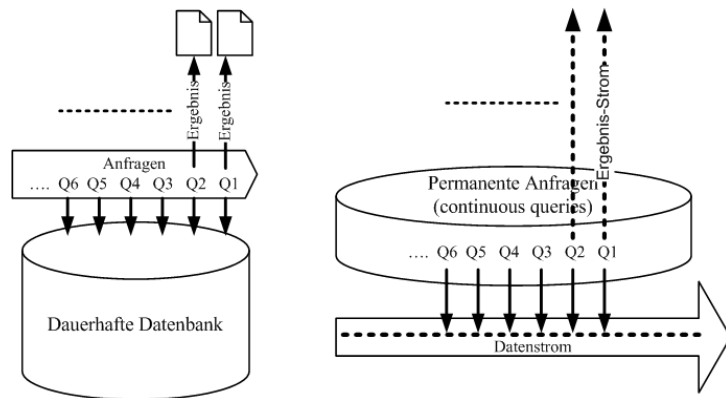


Datenströme

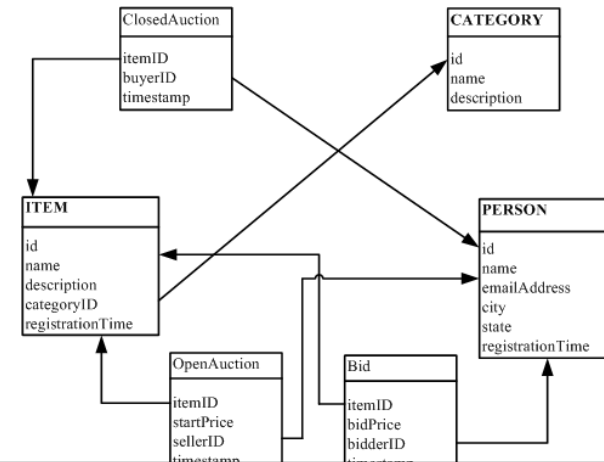


- RFID-Leser, die in der Nähe befindliche („vorbeikommende“) RFID-Tags (Radio Frequency Identifiers) lesen. Diese Sensoren emittieren Tripel der Art (ReaderID, EPC-Code, TimeStamp), wobei der EPC-Code (electronic product code) eine eindeutige Warenkennzeichnung darstellt.
- Umweltsensoren, um Klimadaten zu ermitteln.
- Börsenticker emittieren die jüngsten Aktienkurse basierend auf den letzten Handelstransaktionen.
- Kameras und andere Sensoren liefern kontinuierliche Informationen über bewegliche Objekte wie z.B. Autos, Menschen, etc.
- Fast alle Menschen liefern über ihr Handy kontinuierlich Daten über ihr Bewegungsmuster. Manche Personen machen es den interessierten Unternehmen noch einfacher, ein präzises Bewegungsmuster via GPS-Daten zu erstellen.

Datenbank versus Datenstrom



Beispiel-Datenstrom



Datenstrom-Definition und einfache Anfrage



```
create stream OpenAuction(itemID int, sellerID int,
                          startPrice real, time timestamp)
source establishConnection('port4711', 'converter')
ordered by time;
```

```
select itemID, DollarToEuro(bidPrice), bidderID
from Bid
```

25

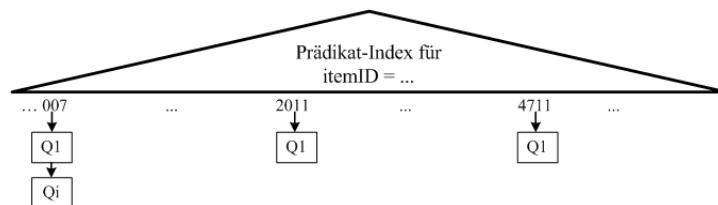
Subskriptions-Anfrage



```
Q1: select *
from Bid
where itemID = 4711 or itemID = 007 or itemID = 2011 or ...
```

26

Prädikat-Index



27

Bereichsanfrage

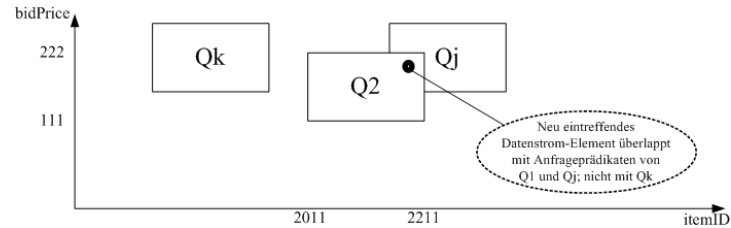


-

```
Q2: select *
from Bid
where itemID between 2011 and 2211 and
bidPrice between 111 and 222
```

28

R-Baum-Index



29

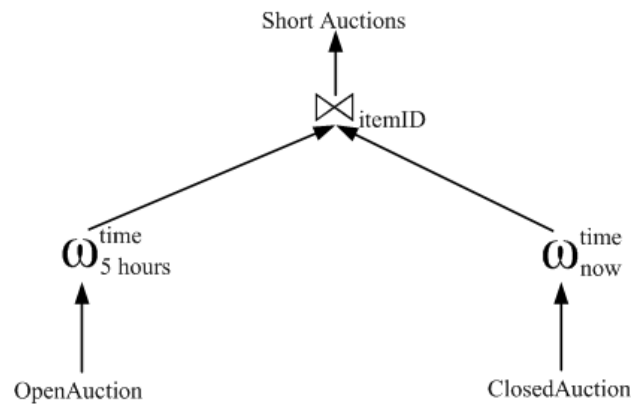
Fenster-Anfrage



```
select O.*
from OpenAuction O window(range 5 hours),
     ClosedAuction c
where o.itemID = c.itemID
```

30

Auswertung: Short Auctions



31

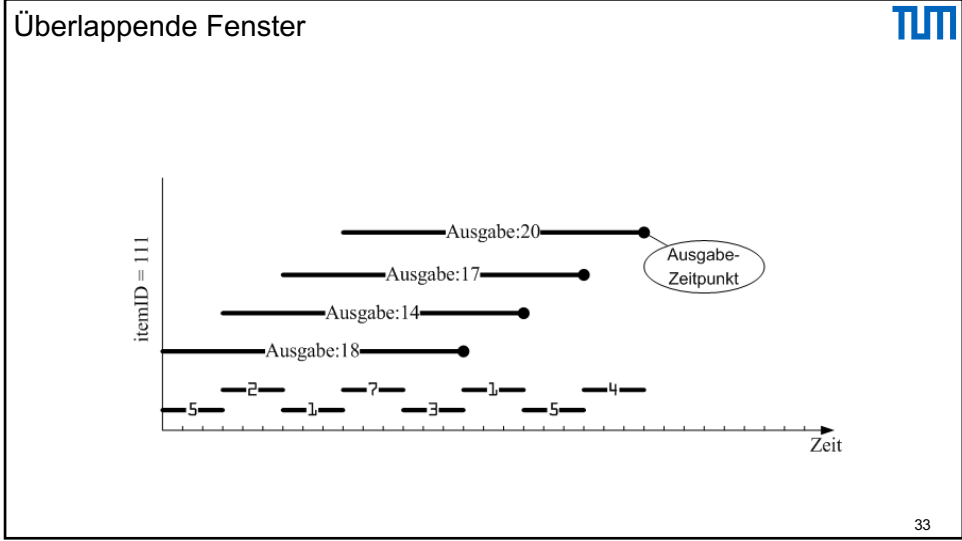
Sliding Windows



```
select itemID, count(*)
from Bid window(range 15 minutes slide 3 minutes)
group by itemID
```

```
select itemID, bidPrice
from Bid b1 window(range 10 minutes)
where b1.bidPrice = (select max(b2.bidPrice)
                    from Bid b2 window(range 10 minutes))
```

32

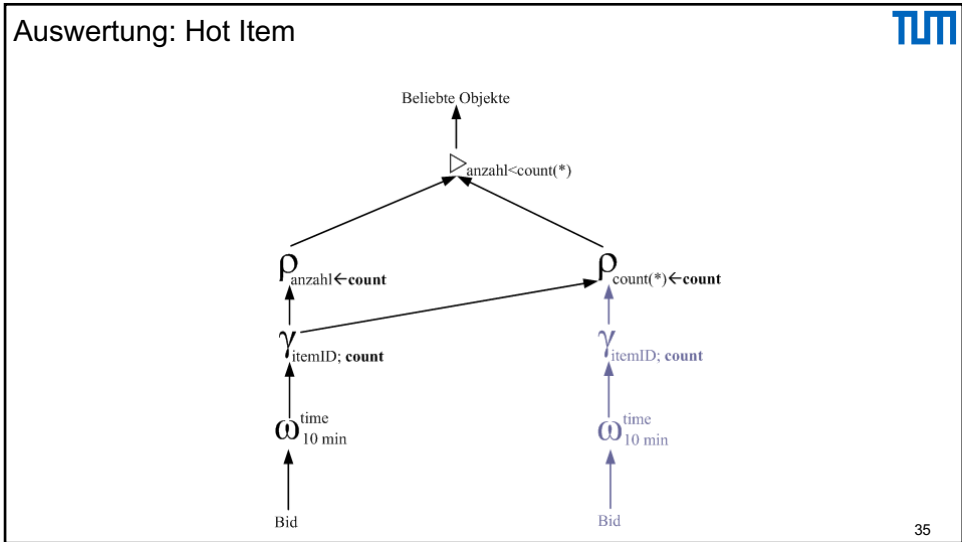


Hot Items-Anfrage

```

select itemID
from (select b1.itemID as itemID, count(*) as anzahl
      from Bid b1 window(range 10 minutes)
      group by b1.itemID)
where anzahl >= all (select count(*)
                    from Bid b2 window(range 10 minutes)
                    group by b2.itemID)
    
```

34



Information Retrieval

Informationsexplosion im Internet

Ranking von Dokumenten um relevante Information zu finden

Ähnlichkeit von Dokumenten (Dissertationen) zu erkennen

36

TF-IDF: Term Frequency – Inverse Document Frequency



$$TF_{ij} = f_{ij} / \sum_{i=1 \dots |V|} f_{ij}$$

$$IDF_i = \log(N/n_i)$$

$$rel(D_j, Q) = \sum_{i \in Q} TF_{ij} * IDF_i$$

Relevanz-Ranking am Beispiel



D_1	D_2	D_3
Nach dem Spiel ist vor dem Spiel.	Was wir ersinnen ist des Zufalls Spiel.	Der Ball ist rund und ein Spiel dauert neunzig Minuten.

Wort i	TF_{ij}			IDF_i			
	D_1	D_2	D_3	N	n_i	N/n_i	$\log(N/n_i)$
1: Ball	0	0	1/3	3	1	3	0,477121255
2: Minute	0	0	1/3	3	1	3	0,477121255
3: Spiel	2/2	1/2	1/3	3	3	1	0
4: Zufall	0	1/2	0	3	1	3	0,477121255

Anfrage $Q \equiv \text{Ball} \wedge \text{Spiel}$ ermittelt man für das Dokument D_3 ,

$$rel(D_3, Q) = 1/3 * 0,477121255 + 1/3 * 0 = 0,1590404182$$

Relevanz-Ranking am Beispiel



D_1	D_2	D_3
Nach dem Spiel ist vor dem Spiel.	Was wir ersinnen ist des Zufalls Spiel.	Der Ball ist rund und ein Spiel dauert neunzig Minuten.

Wort i	TF_{ij}			IDF_i			
	D_1	D_2	D_3	N	n_i	N/n_i	$\log(N/n_i)$
1: Ball	0	0	1/3	3	1	3	0,477121255
2: Minute	0	0	1/3	3	1	3	0,477121255
3: Spiel	2/2	1/2	1/3	3	3	1	0
4: Zufall	0	1/2	0	3	1	3	0,477121255

Für die Anfrage $Q' \equiv \text{Ball} \wedge \text{Spiel} \wedge \text{Zufall}$ hat D_2 den höchsten Relevanzwert, nämlich

$$rel(D_2, Q') = 0 * 0,477121255 + 1/2 * 0 + 1/2 * 0,477121255 = 0,2385606274$$

Invertierte Indexierung

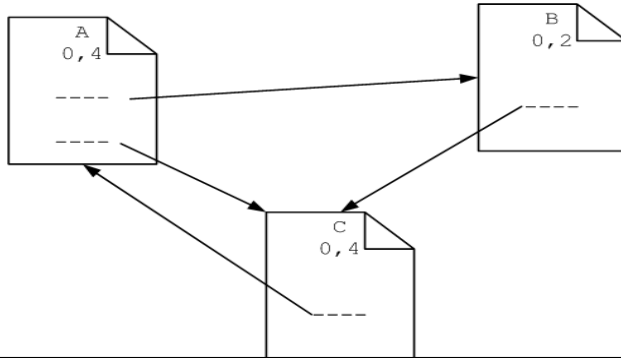


Begriff	→	Dokumente
Ball	→	$(D_3, 1)$
Minute	→	$(D_3, 1)$
Spiel	→	$(D_1, 2) \quad (D_2, 1) \quad (D_3, 1)$
Zufall	→	$(D_2, 1)$

Page Rank: Grundidee



$$r(A) = \frac{\alpha}{N} + (1 - \alpha) \left(\frac{r(B_1)}{|B_1|} + \dots + \frac{r(B_n)}{|B_n|} \right)$$

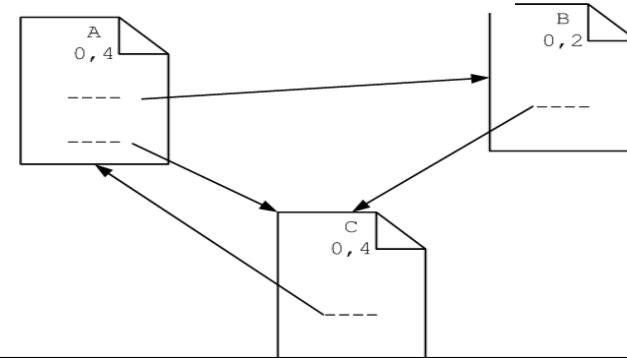


41

Page Rank: Grundidee



$$\begin{aligned} r(A) &= r(C)/1 \\ r(B) &= r(A)/2 \\ r(C) &= r(A)/2 + r(B) \end{aligned}$$



42

Mathematisches Modell des PageRank



$$M_{ij} = \begin{cases} 1/|P_j| & \text{falls } P_j \text{ auf } P_i \text{ verweist} \\ 0 & \text{sonst} \end{cases}$$

$$M = \begin{pmatrix} 0 & 0 & 1 \\ 1/2 & 0 & 0 \\ 1/2 & 1 & 0 \end{pmatrix} \quad p_0 = \begin{pmatrix} 1/3 \\ 1/3 \\ 1/3 \end{pmatrix}$$

Man berechnet dann iterativ die Vektoren

$$p_1 = M * p_0, p_2 = M * p_1 = M * (M * p_0) = M^2 * p_0, \dots, p_i = M^i * p_0$$

43

Mathematisches Modell des PageRank: unser Beispiel



$$M = \begin{pmatrix} 0 & 0 & 1 \\ 1/2 & 0 & 0 \\ 1/2 & 1 & 0 \end{pmatrix} \quad p_0 = \begin{pmatrix} 1/3 \\ 1/3 \\ 1/3 \end{pmatrix}$$

$$p_1 = M p_0 = \begin{pmatrix} 0 & 0 & 1 \\ 1/2 & 0 & 0 \\ 1/2 & 1 & 0 \end{pmatrix} * \begin{pmatrix} 1/3 \\ 1/3 \\ 1/3 \end{pmatrix} = \begin{pmatrix} 1/3 \\ 1/6 \\ 1/2 \end{pmatrix}$$

$$p_7 = M^7 * p_0 = \begin{pmatrix} 20/48 \\ 9/48 \\ 19/48 \end{pmatrix} \quad p_2 = M * M p_0 = M * p_1 = \begin{pmatrix} 1/2 \\ 1/6 \\ 1/3 \end{pmatrix}$$

44

Konvergenz und Dämpfung

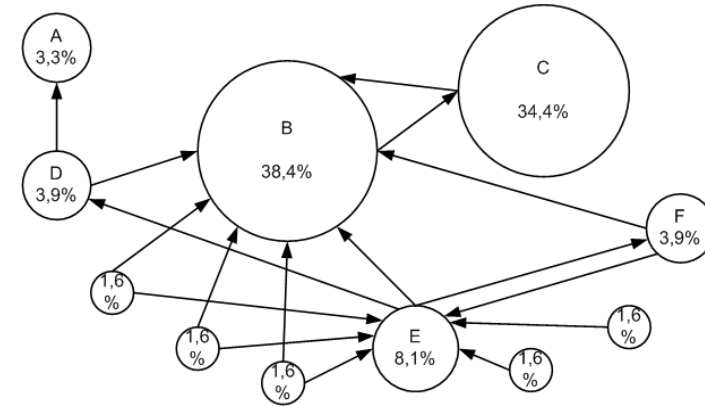


$$p_\infty = \lim_{n \rightarrow \infty} M^n p_0$$

$$p_i = (((1 - \alpha) * M) * p_{i-1}) + \begin{pmatrix} \frac{\alpha}{N} \\ \vdots \\ \frac{\alpha}{N} \end{pmatrix}$$

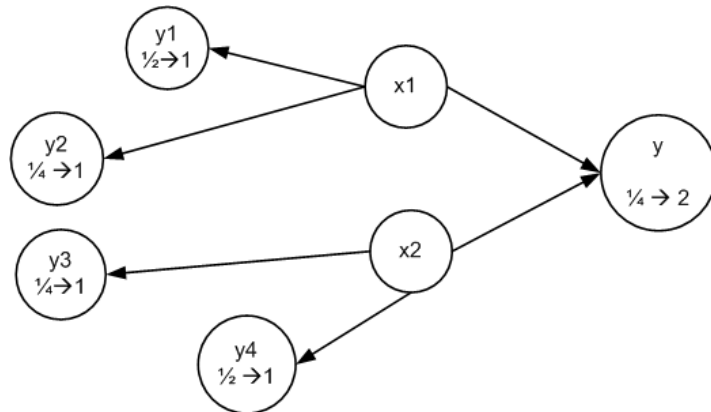
45

PageRank für größeren Graph [aus Wikipedia]



46

HITS-Algorithmus: Hubs und Autoritäten



47

HITS-Algorithmus: Hubs und Autoritäten



Dann wird der Hub-Wert einer Seite i wie folgt definiert:

$$h_i = \delta \sum_{j=1 \dots N} A_{ij} a_j$$

Analog wird die Gewichtung der Autorität einer Seite i wie folgt errechnet:

$$a_i = \lambda \sum_{k=1 \dots N} A_{ik}^T h_k$$

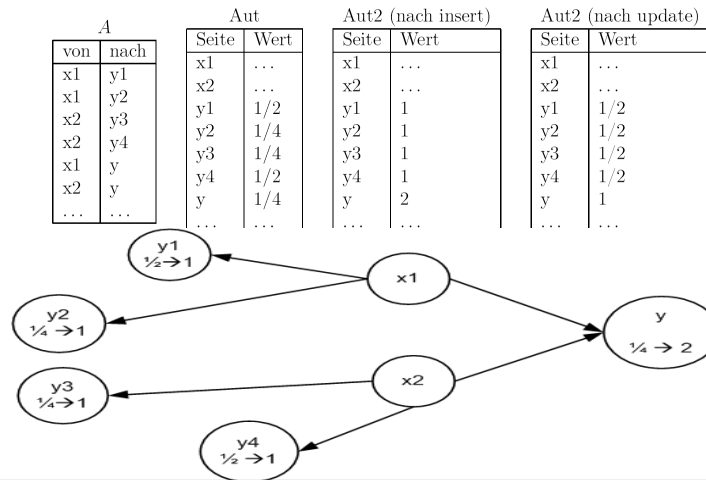
Eigenvektoren der Matrizen AA^T bzw. $A^T A$

$$h = \delta \lambda A A^T h$$

$$a = \delta \lambda A^T A a$$

48

Relationale HITS-Modellierung



49

Algorithmus



1. Berechne die Hub-Werte jeder Seite q indem man die Summe der Autoritäts-werte aller Seiten r ermittelt, auf die q verweist.
2. Berechne die Autorität der Seite p durch Summierung der Hub-Werte der Seiten q , die auf p verweisen
3. Normalisiere die so erhaltenen Autoritätswerte indem man sie mit $\lambda = 1/\max$ multipliziert, wobei \max den Maximalwert aller gerade neu berechneten Autoritätswerte darstellt. Diese Normalisierung ist nötig, um die Werte nicht ins Unermessliche steigen zu lassen.

50

Algorithmus ... in SQL



```

insert into Aut2 (
  select a1.nach, sum(Aut.Wert)
  from Aut, A a1, A a2
  where Aut.Seite = a2.nach and a1.von = a2.von
  group by a1.nach )

update Aut2
  set Wert = Wert / (select max(Wert) from Aut2)

```

51

Graph Exploration



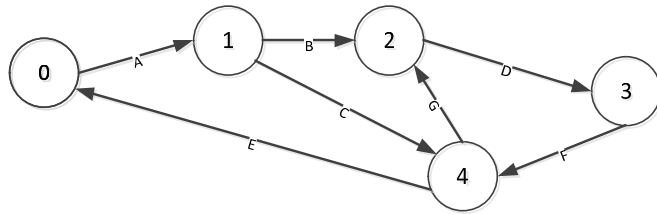
Analyse großer sozialer Netzwerke (Facebook, Twitter, Instagram, etc)

Analyse von Kommunikationsnetzen

...

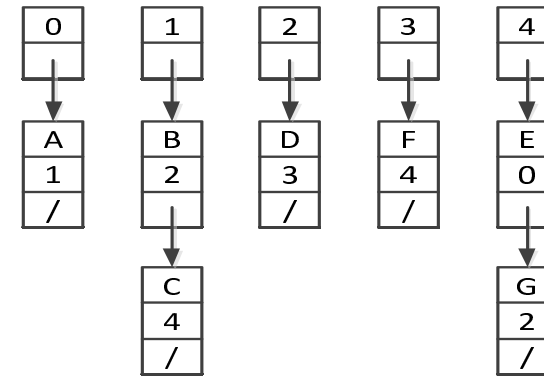
52

Darstellung von Graphen



53

Adjazenzlisten-Darstellung



: Die Darstellung des Graphen als Adjazenzliste

54

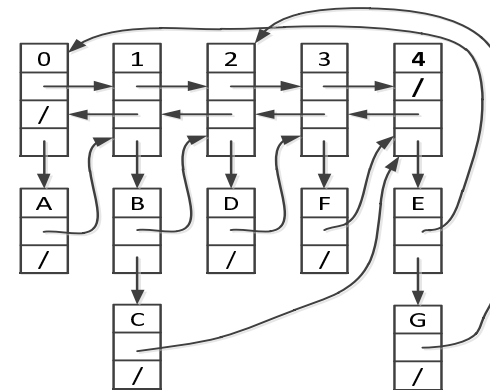
Adjazenzmatrix



$$A = \begin{pmatrix} 0 & A & 0 & 0 & 0 \\ 0 & 0 & B & 0 & C \\ 0 & 0 & 0 & D & 0 \\ 0 & 0 & 0 & 0 & F \\ E & 0 & G & 0 & 0 \end{pmatrix}$$

55

Verzweigte Adjazenzlisten-Struktur



Die vollständig verzweigte Adjazenzlisten-Struktur

56

Compressed sparse rows (CSR)



$$\text{Werte} = \begin{pmatrix} A & B & C & D & F & E & G \\ (0,1) & (1,2) & (1,4) & (2,3) & (3,4) & (4,0) & (4,2) \end{pmatrix}$$

$$\text{SpaltenIndex} = \begin{pmatrix} 1 & 2 & 4 & 3 & 4 & 0 & 2 \\ (0) & (1) &) & (2) & (3) & (4) &) \end{pmatrix}$$

$$\text{ZeilenPtr} = \begin{pmatrix} 0 & 1 & 3 & 4 & 5 & 7 \\ (<0) & (0..0) & (0..1) & (0..2) & (0..3) & (0..4) \end{pmatrix}$$

$$A_{ij} = \text{Wert}[k] \iff \text{SpaltenIndex}[k] = j \wedge \text{ZeilenPtr}[i] \leq k < \text{ZeilenPtr}[i+1]$$

57

Rekonstruktion aus der CSR



$$A_{ij} = \text{Wert}[k] \iff \text{SpaltenIndex}[k] = j \wedge \text{ZeilenPtr}[i] \leq k < \text{ZeilenPtr}[i+1]$$

Die Rekonstruktion – beispielsweise der Zeile 1 der Matrix A – geschieht dann wie folgt: Die Einträge 1 bzw. 3 an den Positionen 1 und 2 des Vektors *ZeilenPtr* zeigen an, dass die Zeile 1 insgesamt $3 - 1 = 2$ nicht-Null-Einträge hat. Diese Adjazenzmatrix-Werte findet man also an den Positionen 1 und 2 des *Werte*-Vektors, also B und C . Anhand des *SpaltenIndex*-Vektors ermittelt man dann noch, dass sie zu den Spalten 2 und 4 gehören, also ergibt sich $A_{12} = B$ und $A_{14} = C$. Für die automatische Rekonstruktion baut man sich natürlich einen entsprechenden Iterator, der die nicht-Null-Einträge der Adjazenzmatrix mitsamt ihren Zeile/Spalte-Indizes der Reihe nach zurückgibt. Dies sei den Lesern als Übung empfohlen.

58

Zentralitätsmaße



aus den Sozialwissenschaften

charakterisiert ganze Graphen oder Teilstrukturen

Sinnhaftigkeit ist manchmal zweifelhaft

59

Verbindungscentralität



(degree centrality)

$$C_D(v) = \text{degree}(v)$$

$$C_D(G) = \sum_{i=1}^{|V|} [C_D(v^*) - C_D(v_i)]$$


60

Normierung mit dem Maximalwert für den sternförmigen Graphen 

$$C_D(G^*) = \sum_{i=1}^{|V|} [C_D(v^*) - C_D(v_i)] = (|V| - 2)(|V| - 1)$$

$$C'_D(G) = C_D(G) / [(|V| - 2)(|V| - 1)]$$

61


Nähe-Zentralität (closeness centrality) 

$$C(v) = \frac{1}{\sum_{y \in V} d(y, v)}$$

$$H(v) = \sum_{v \neq y \in V} 1/d(y, v)$$

Hierbei definiert man $1/\infty$ als 0.

62

Nähe-Zentralität (closeness centrality) 

stärkere Gewichtung der Nähe zweier Knoten:

$$D(v) = \sum_{v \neq y \in V} \frac{1}{2^{d(y, v)}}$$

63

Pfad-Zentralität (betweenness centrality) 

Für einen Knoten v im Graph $G = (V, E)$ wird also dieser Wert wie folgt bestimmt:

1. Für jedes Knotenpaar (s, t) berechne deren kürzeste Pfade im Graphen.
2. Bestimme die Anzahl der kürzesten Pfade von s nach t als σ_{st} .
3. Für jedes Knotenpaar (s, t) bestimme die Anzahl der kürzesten Pfade, die durch den betrachteten Knoten v verlaufen. Dieser Wert sei als $\sigma_{st}(v)$ bezeichnet.

$$C_B(v) = \sum_{s \neq v \neq t \in V} (\sigma_{st}(v) / \sigma_{st})$$

64

Pfad-Zentralität (betweenness centrality)



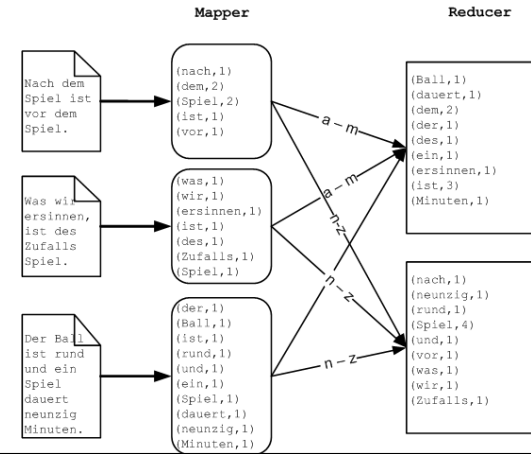
Normierung mit dem Wert für den zentralsten Knoten in einem sternförmigen Graphen:

$$C_B(v) = \sum_{s \neq v \neq t \in V} (\sigma_{st}(v) / \sigma_{st})$$

$$C_B^*(v) = C_B(v) / [((n-1)(n-2))/2] = \frac{\sum_{s \neq v \neq t \in V} (\sigma_{st}(v) / \sigma_{st})}{[(n-1)(n-2)]/2}$$

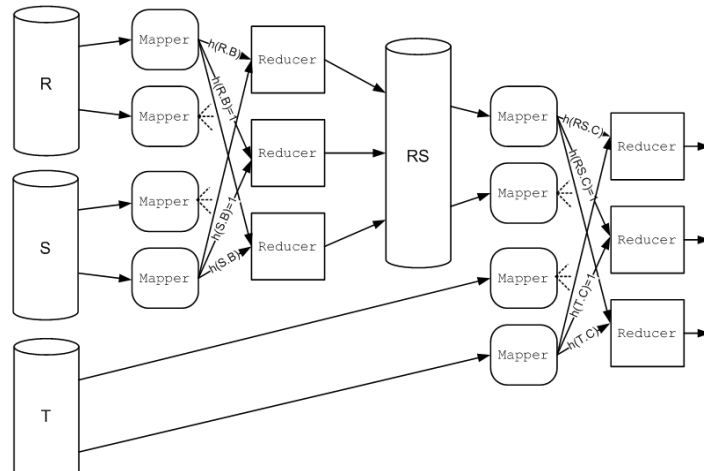
65

Map Reduce



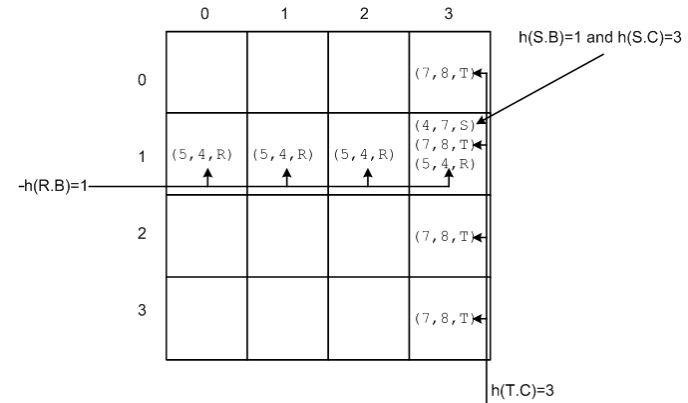
66

Join mit Map Reduce



67

Verbesserung nach Ullman



68

Map Reduce Skriptsprache: PigLatin

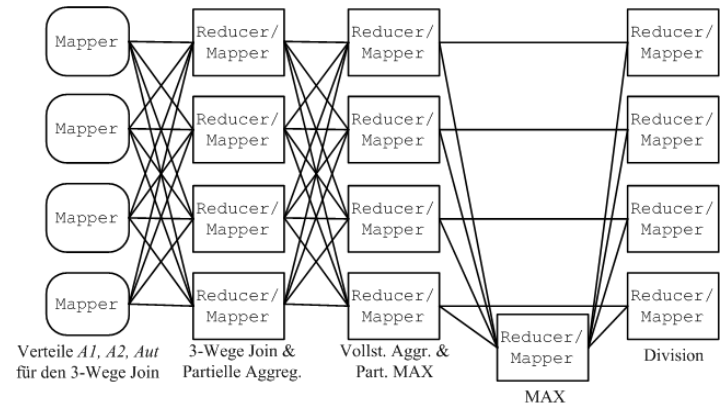


```

a1 = LOAD 'A' AS (a1von, a1nach);
a2 = LOAD 'A' AS (a2von, a2nach);
aut = LOAD 'Aut' AS (seite, wert);
j1 = JOIN a2 BY a2nach, aut BY seite;
j2 = JOIN a1 BY a1von, j1 BY a2von;
g1 = GROUP j2 BY a1nach;
aut2 = FOREACH g1 GENERATE group AS seite, SUM(j2.wert) AS wert;
g2 = GROUP aut2 ALL;
max = FOREACH g2 GENERATE MAX(aut2.wert) AS max;
c = CROSS aut2, max;
aut2Up = FOREACH c GENERATE seite, wert / max;
STORE aut2Up INTO 'Aut2';
    
```

69

Auswertung des HITS Algorithmus



70

Peer to Peer-Informationssysteme

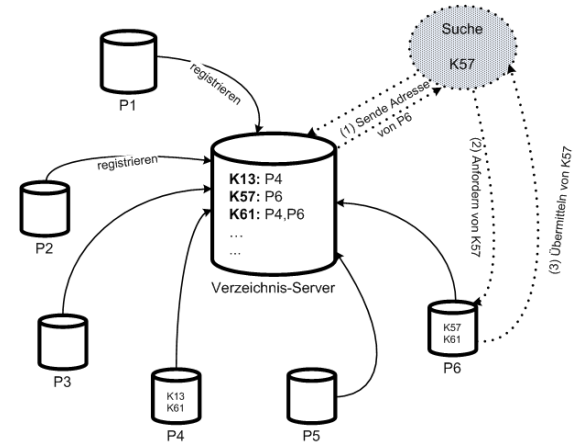


Seti@Home
 • P2P number crunching

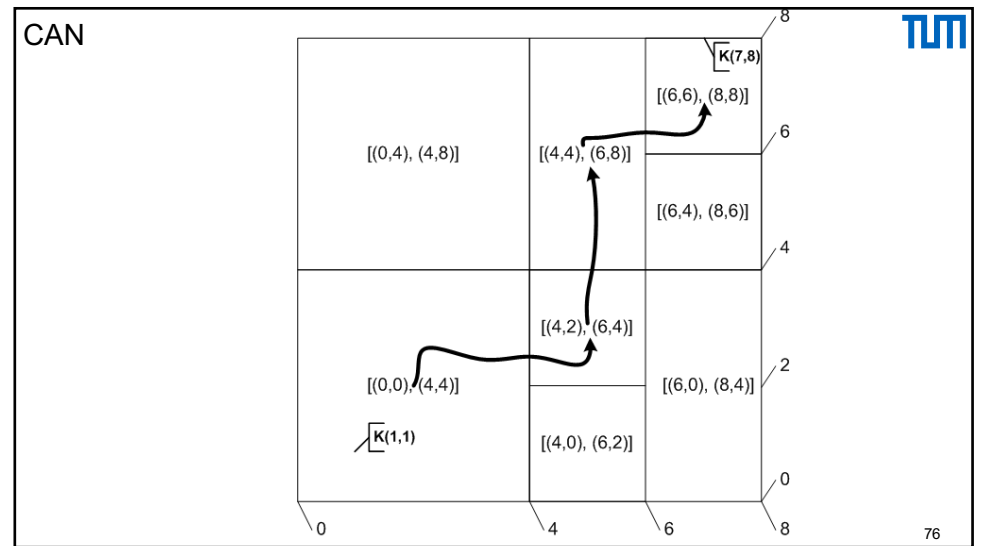
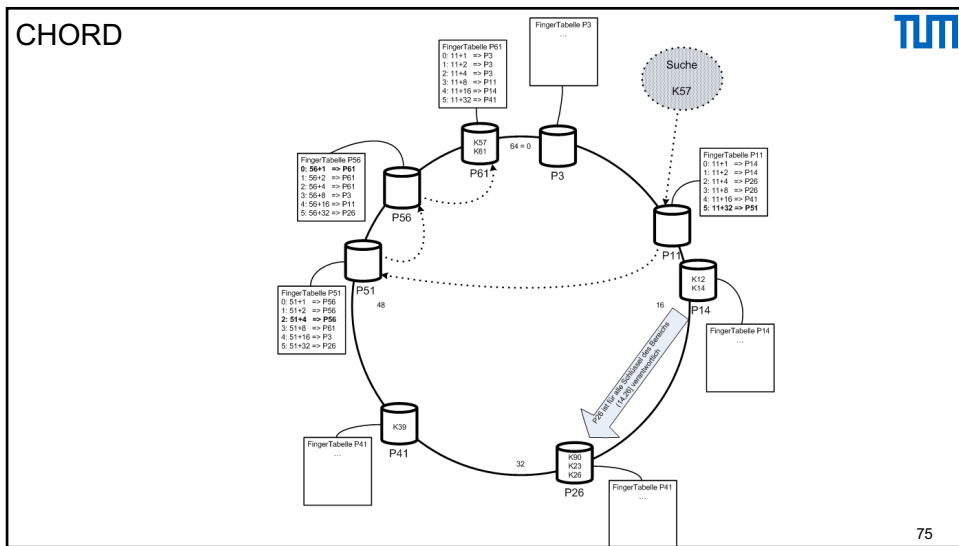
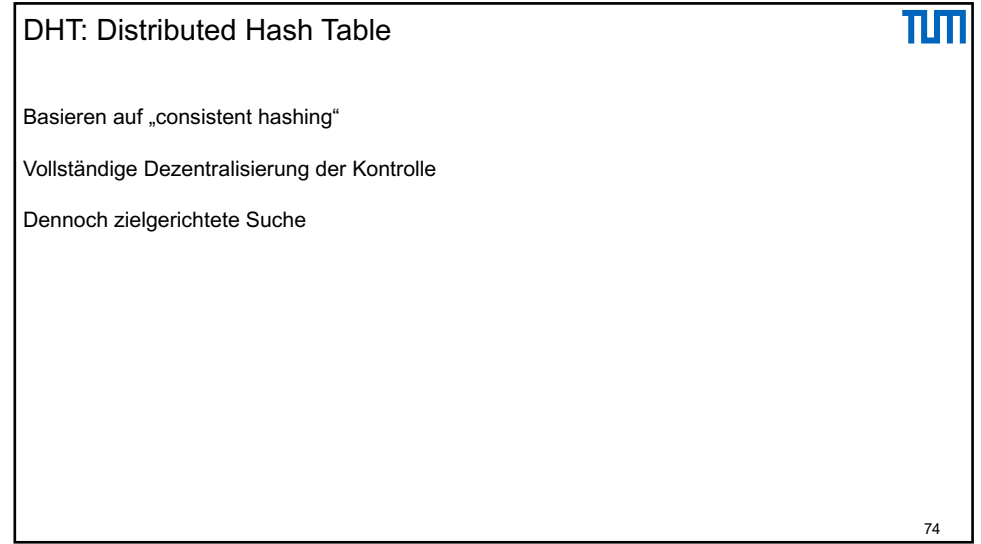
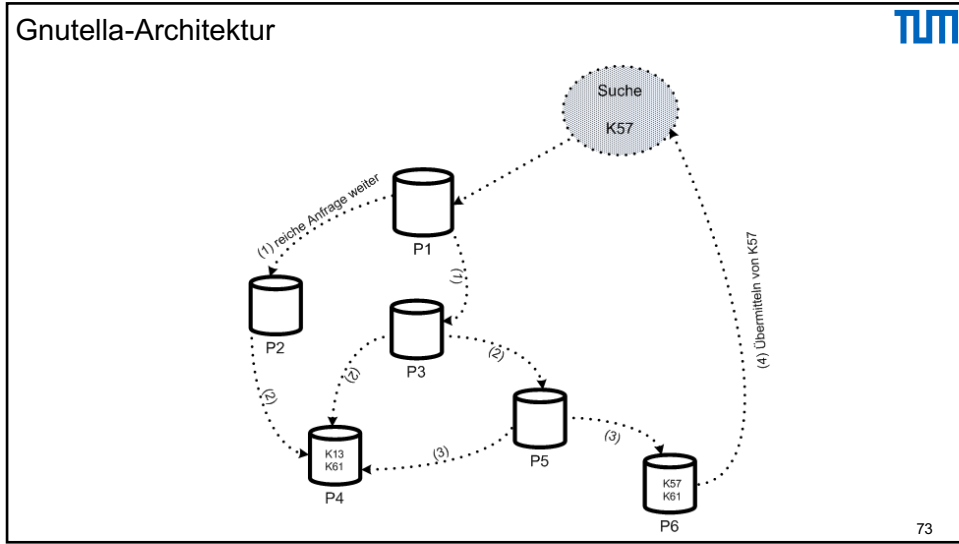
Napster
 • P2P file sharing / Informationsmanagement

71

Napster-Architektur



72



No-SQL Datenbanken



Internet-scale Skalierbarkeit

CAP-Theorem: nur 2 von 3 Wünschen erfüllbar

- Konsistenz (Consistency)
- Zuverlässigkeit/Verfügbarkeit (Availability)
- Partitionierungs-Toleranz

No-SQL Datenbanksysteme verteilen die Last innerhalb eines Clusters/Netzwerks

- Dabei kommen oft DHT-Techniken zum Einsatz

77

Schnittstelle der No-SQL Datenbanken



Insert(k,v)

Lookup(k)

Delete(k)

Extrem einfach → effizient

Aber: wer macht denn die Joins/Selektionen/...

- → das Anwendungsprogramm

78

Konsistenzmodell: CAP



Relaxiertes Konsistenzmodell

- Replizierte Daten haben nicht alle den neuesten Zustand
 - Vermeidung des (teuren) Zwei-Phasen-Commit-Protokolls
- Transaktionen könnten veraltete Daten zu lesen bekommen
- Eventual Consistency
 - Würde man das System anhalten, würden alle Kopien irgendwann (also eventually) in denselben Zustand übergehen
- Read your Writes-Garantie
 - Tx leist auf jeden Fall ihre eigenen Änderungen
- Monotonic Read-Garantie
 - Tx würde beim wiederholten Lesen keinen älteren Zustand als den vorher mal sichtbaren lesen

79

Systeme



MongoDB

Cassandra

Dynamo

BigTable

Hstore

SimpleDB

S3

80

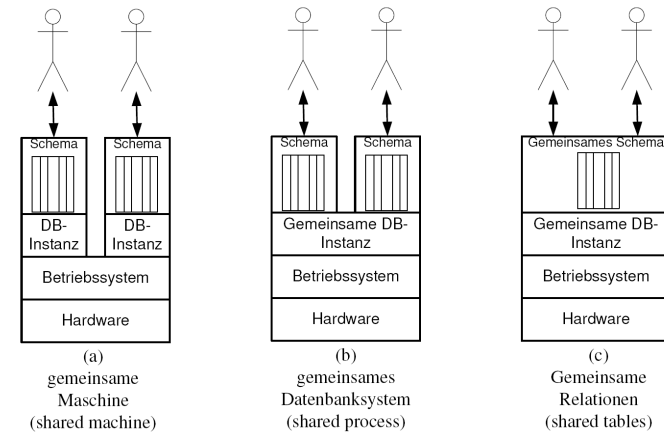
Multi-Tenancy / Cloud-Datenbanken



- **Infrastructure as a Service (IaaS):** Hierbei werden den Kunden de facto virtuelle Maschinen zur Verfügung gestellt, auf denen dann beliebige Software installiert werden könnte. Kunden könnten also auch existierende Anwendungen auf eine derartige virtuelle Maschine portieren. Insbesondere kann man natürlich „normale“ Datenbanksysteme installieren. Amazon Web Services ist ein typisches Beispiel einer IaaS, die aber zusätzlich auch nicht-relationale (also No-SQL) Datenbankfunktionalität durch die Systeme SimpleDB und S3 anbietet.
- **Platform as a Service (PaaS):** Hierzu zählen die Systeme Google AppEngine und Microsoft Azure, die reichhaltige Schnittstellen für die Neuentwicklung von Web-Applikationen bereitstellen. Hierzu dienen insbesondere die Datenspeicher, also Google's App Engine Datastore oder Microsoft's Azure Table Storage.
- **Software as a Service (SaaS):** Diese Systeme stellen komplexe, anwendungsspezifische Funktionalität zur Verfügung. Bekannteste Beispiele im betrieblichen Umfeld sind das Customer-Relationship-Management-System von Salesforce oder das umfassende betriebliche Anwendungssystem Business-By-Design von SAP. Diese Software wird nicht mehr bei den Nutzern installiert sondern von den Betreibern als „Hosting-Modell“ angeboten und von den Nutzern via Web-Schnittstellen genutzt.

81

Multi-Tenancy Datenbankarchitekturen



82

Shared Tables



Account			
Tenant	Aid	Name	...
17	1	Acme	...
17	2	Gump	...
35	1	Ball	...
42	1	Big	...

Die Applikationssoftware muss dann sicherstellen, dass jede Anfrage entsprechend umgeformt wird, je nachdem von welchem Nutzer sie empfangen wurde:

```
Tenant 35 → select * from Account where Name like 'B%'
           → select * from Account where Name like 'B%' and Tenant = 35
```

83

Private Relationen



Account17			
Aid	Name	Hospital	Beds
1	Acme	St. Mary	135
2	Gump	State	1042

Account35	
Aid	Name
1	Ball

Account42		
Aid	Name	Dealers
1	Big	65

84

Erweiterungs-Relationen



Account_Ext			
Tenant	Row	Aid	Name
17	0	1	Acme
17	1	2	Gump
35	0	1	Ball
42	0	1	Big

Healthcare_Account			
Tenant	Row	Hospital	Beds
17	0	St. Mary	135
17	1	State	1042

Automotive_Account		
Tenant	Row	Dealers
42	0	65

85

Universal Relation



Universal							
Tenant	Table	Col1	Col2	Col3	Col4	Col5	eCol6
17	0	1	Acme	St. Mary	135	-	-
17	0	2	Gump	State	1042	-	-
35	1	1	Ball	-	-	-	-
42	2	1	Big	65	-	-	-

86

Zerlegung: Pivot-Relationen



Pivot_int				
Tenant	Table	Col	Row	Int
17	0	0	0	1
17	0	3	0	135
17	0	0	1	2
17	0	3	1	1042
35	1	0	0	1
42	2	0	0	1
42	2	2	0	65

Pivot_str				
Tenant	Table	Col	Row	Str
17	0	1	0	Acme
17	0	2	0	St. Mary
17	0	1	1	Gump
17	0	2	1	State
35	1	1	0	Ball
42	2	1	0	Big

87

Ballung logisch verwandter Werte: Chunk Tables



Account_Row			
Tenant	Row	Aid	Name
17	0	1	Acme
17	1	2	Gump
35	0	1	Ball
42	0	1	Big

Chunk_Row						
Name	Tenant	Table	Chunk	Row	Int1	Str1
Acme	17	0	0	0	135	St. Mary
Gump	17	0	0	1	1042	State
Ball	42	2	0	0	65	-
Big						

88

Key/Value-Store



HBase Key/Value-Store		
Row Key	Account	Contact
17Act1	[name:Acme, hospital:St. Mary, beds:135]	
17Act2	[name:Gump, hospital:State, beds:1042]	
17Ctc1		[...]
17Ctc2		[...]
35Act1	[name:Ball]	
35Ctc1		[...]
42Act1	[name:Big, dealers:65]	

89

XML-basiertes Schema



Account			
Tenant	Aid	Name	Ext_XML
17	1	Acme	<pre><ext> <hospital>St. Mary</hospital> <beds>135</beds> </ext></pre>
17	2	Gump	<pre><ext> <hospital>State</hospital> <beds>1042</beds> </ext></pre>
35	1	Ball	
42	1	Big	<pre><ext> <dealers>65</dealers> </ext></pre>

90