Hochschule RheinMain

NO SQL

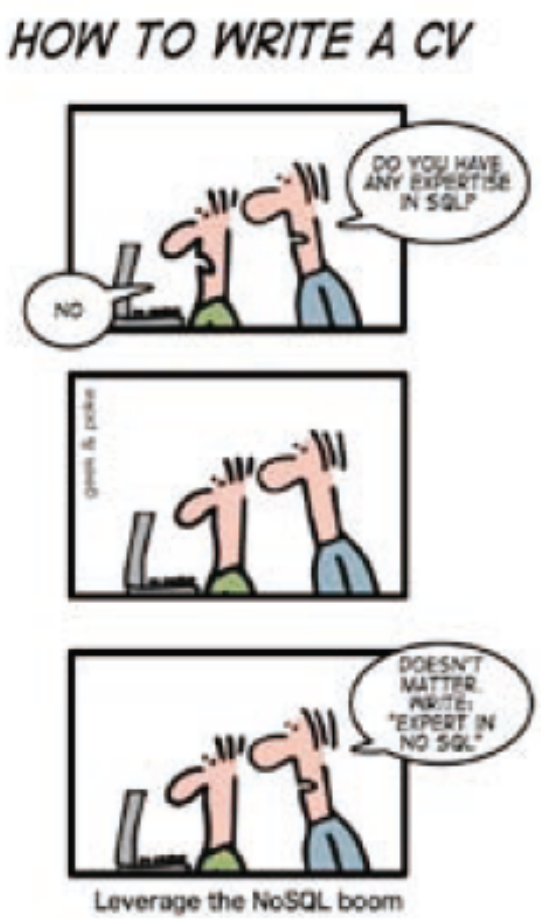
Einstieg in die Welt der nichtrelationalen Datenbanken

Dr. Eva-Maria Iwer
Mai/ Juni 2022

Hochschule RheinMain

WAS IST NOT ONLY SQL

Ein kurzer Überblick



Quelle: http://geekandpoke.typepad.com/geekandpoke/2011/01/nosql.html

DEFINITION

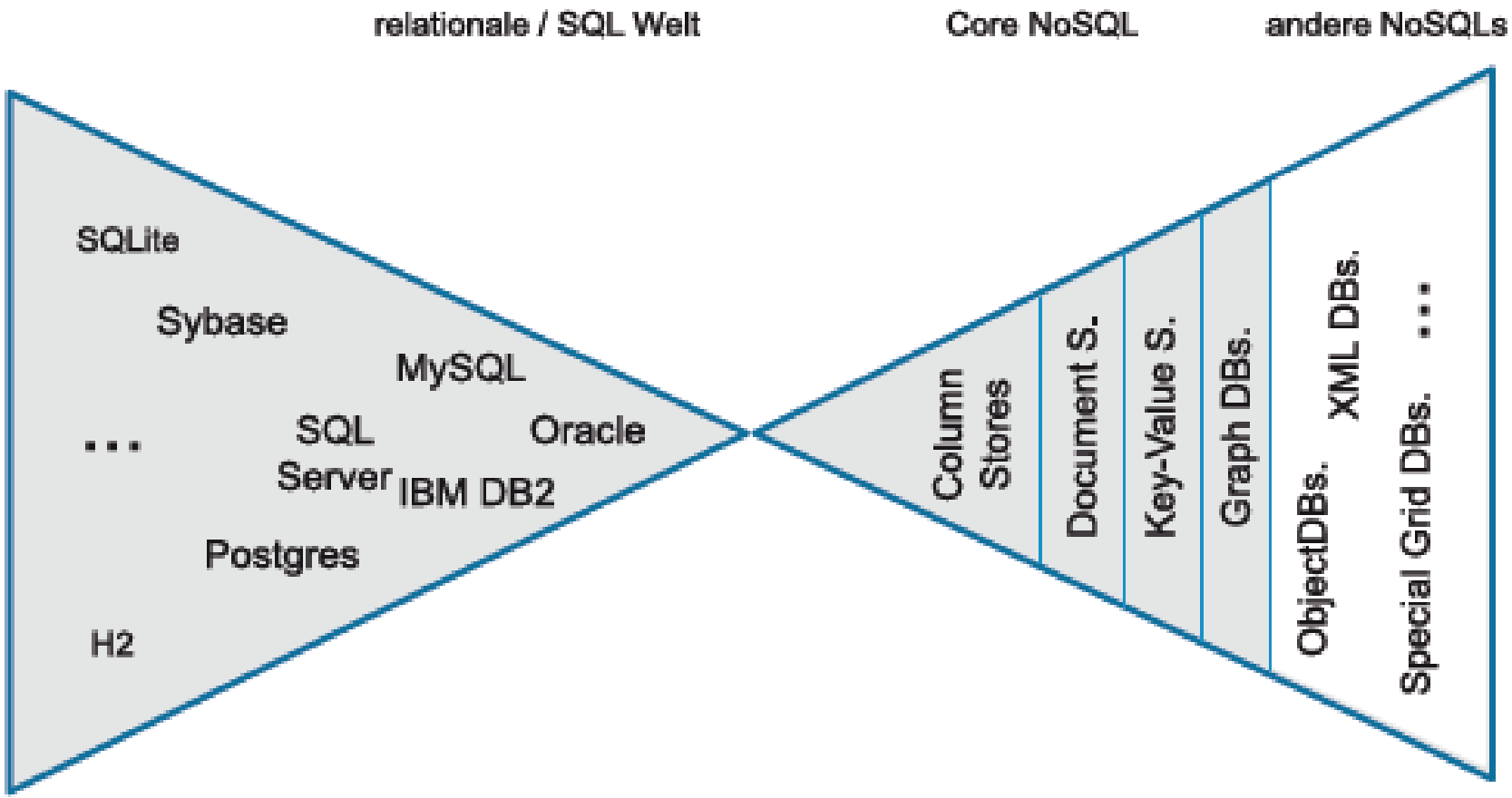
Hochschule RheinMain

Unter NoSQL wird eine neue Generation von Datenbanksystemen verstanden, die meistens einige der nachfolgenden Punkte berücksichtigen:

1. Das zugrundeliegende Datenmodell ist nicht relational.
2. Die Systeme sind von Anbeginn an auf eine verteilte und horizontale Skalierbarkeit ausgerichtet.
3. Das NoSQL-System ist Open Source.
4. Das System ist schemafrei oder hat nur schwachere Schemarestriktionen.
5. Aufgrund der verteilten Architektur unterstutzt das System eine einfache Datenreplikation.
6. Das System bietet eine einfache API.
7. Dem System liegt meistens auch ein anderes Konsistenzmodell zugrunde: *Eventually Consistent* und BASE, aber nicht ACID

KATEGORISIERUNG
Übersicht

Hochschule RheinMain



KATEGORISIERUNG
Key/Value-Systeme

Hochschule RheinMain

- Diese Gruppe von Systemen bietet mindestens ein einfaches Schema aus Schlüssel und Wert an.
- Meistens können jedoch die Schlüssel in Namensraume und Datenbanken aufgeteilt werden. Auch die Values sind meistens nicht nur Zeichenketten, häufig können auch Hashes, Sets oder Listen Values sein.
- viele dieser Systeme ähnelnd der Gruppe der Column-Family-Systeme.
- **Vorteil der Key/Value-Systeme**
  - ist das einfache Datenmodell, das meistens auch eine schnellere und effizientere Datenverwaltung zur Folge hat.
- **Nachteile:**
  - Abfragemächtigkeit lässt oft zu wünschen übrig lasst
- Eigene, beliebig komplexe Queries können oft nicht selbst geschrieben werden, stattdessen muss man sich oft auf die Mächtigkeit der API verlassen.
- Klassiker: Amazon-Systeme (Dynamo und S3), Redis, Voldemort und Scalaris

KATEGORISIERUNG
Column-Family-Systeme

Hochschule RheinMain

- Zu den typischen Vertretern dieser Familie zahlen HBase, Cassandra und Hypertable.
- Die Datenstrukturen ähneln manchmal Excel-Tabellen, haben aber dennoch gewichtige Unterschiede.
- beliebige Schlüssel können auf beliebig viele Key/Value-Paare angewendet werden
- Spalte kann mit beliebigen Key/Value-Paaren erweitert werden.
- Daher auch der Name Column Family.
- Viele Systeme bieten auch Super-Columns in Form von Sub-Listen an. Die Keys- und Value-Listen können dann meistens nochmals in Form von Keyspaces oder Clustern organisiert werden.
- Mischung aus Key/Value-System und spaltenorientierter Datenbank oder teilweise auch an relationale Datenbanken.

KATEGORISIERUNG
Document Stores

Hochschule RheinMain

- Document Stores sind im eigentlichen Sinne keine echten Dokumentendatenbanken
- Der Begriff selbst stammt noch aus der Zeit von Lotus Notes, wo tatsachlich echte Anwenderdokumente gespeichert wurden.
- Gemeint sind hier aber nicht Word- oder Textdateien, sondern strukturierte Datensammlungen wie JSON, YAML oder RDF-Dokumente.
- Document Stores legen z.B. JSON-Dateien zusammen mit einer ID ab.
- Meist legt die Datenbank nur fest, auf welches Format die ID weist. Mehr aber auch nicht.
- Couch-DB, MongoDB oder Riak sind die wichtigsten Systeme
- CouchDB und Riak speichern dabei JSON und MongoDB speichert BSON, welches das binare Format von JSON ist.

Hochschule RheinMain

Dazu ein Beispiel:

```
SurName="Doe"
FirstName="John"
  Age="42'
```

als Textdatei wäre ebenfalls ein gültiges Dokument für einen Document Store.

```
{
 'SurName' : "Doe"
 'FirstName" : "John"
   "Age" : 42
}
```

KATEGORISIERUNG
Graphdatenbanken

Hochschule RheinMain

- Verwaltung von Graph- oder Baumstrukturen, in denen die Elemente miteinander verknüpft sind
- Theorie der Graphen recht umfangreich
- Graphendatenbanken haben in den 80er und 90er das Licht der Welt erblickt und wurden zur Modellierung und Verwaltung von Netzen eingesetzt
- Um die Jahrtausendwende gab es aufgrund der Semantic-Web-Forschung immer mehr Systeme in diesem Bereich.
- Und seit 2008 erfahren Graphdatenbanken aufgrund der sogenannten *Location Based Services* (LBS) immer mehr Aufmerksamkeit. Da diese besonders in Smartphones immer mehr Einzug halten, entstand großer Bedarf, Webinformation (z.B. wer sind meine Freunde) mit Geodaten zu verknüpfen.
- Beispiele: Neo4j oder die SonesDB

KATEGORISIERUNG
Vergleich

Hochschule RheinMain

Relationale DB	Server	Database	Table	Primary Key			
Key Value DB	Cluster	Keyspace		Key	Value		
Column Family DB	Cluster	Table/ Keyspace	Column Family	Key	Column Name	Column Value	Super Column optional
Document DB	Cluster	Docspace		Doc Name	Doc Content		
GraphDB	Server	Graphspace	Nodes & Links				

POLYGLOT PERSISTENCE

Mix von DB-Systemen und Auswahl nach Anwendungsart

Hochschule RheinMain

- Die NoSQL-Bewegung sieht sich ein wenig als Vorreiter, um für eine freie Datenbankauswahl zu kämpfen.
- Nicht selten bestehen in der Industrie Rahmenvertrage oder langfristige Bindungen mit bestimmten Datenbankherstellern. Und obwohl im Unternehmen oft sogar Hunderte von verschiedenen Datenbanken mit den unterschiedlichsten Anforderungen erstellt werden müssen, muss dies mit der einen definierten Unternehmensdatenbank geschehen.
- In der Regel ist aber für solche Extremfälle eine Lösung besser, bei der eine vielfaltigere Persistenz (Polyglot Persistence) realisiert wird. Das bedeutet z.B., für Graphen auch Graphdatenbanken einzusetzen.

POLYGLOT PERSISTENCE

Hochschule RheinMain

- Weiter besteht die Idee von NoSQL auch darin, das Bewusstsein für ein großes Datenbankspektrum zu schärfen.
- Viele Entwickler kennen – meistens genau – eine Datenbank besonders gut. Oder in vielen Unternehmen liegt Spezialwissen für genau eine Datenbank vor.

POLYGLOT PERSISTENCE

Hochschule RheinMain

- mehr Arbeit in der Analyse der Daten und der Anforderungen oder Requirements zu machen
- Die Erfahrung zeigt, dass in der Praxis leider viel zu selten untersucht wird, welches Datenmodell wirklich gebraucht wird.
- Und genauso selten wird untersucht, welche Datenbank oder Datenbankkombination wirklich gebraucht wird.

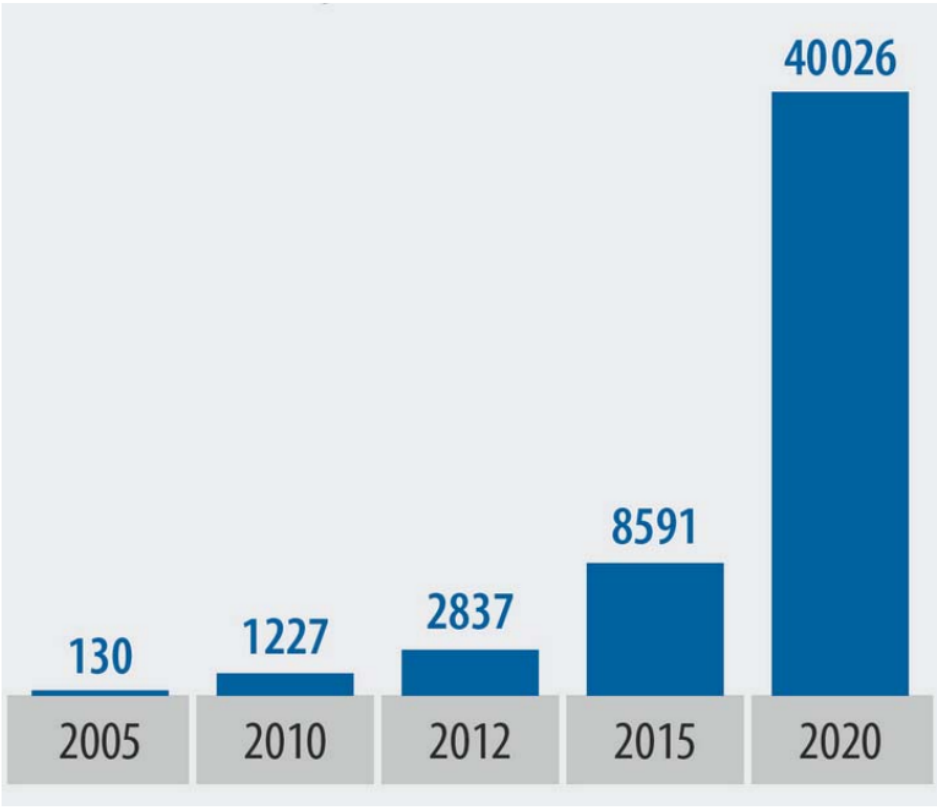
Hochschule RheinMain

BIG DATA

WIE GROß IST BIG DATA

Hochschule RheinMain

- Big wandelt sich schnell
  - Gigabytes,
  - Terabytes (10^{12})
  - Petabytes (10^{15}),
  - Exabytes (10^{18}),
  - Zettabytes (10^{21}),
  - Yottabytes (10^{24}),
  - Brontobytes (10^{27}),
  - …
- bis 2020 werden ca 40 ZB jährlich erzeugt



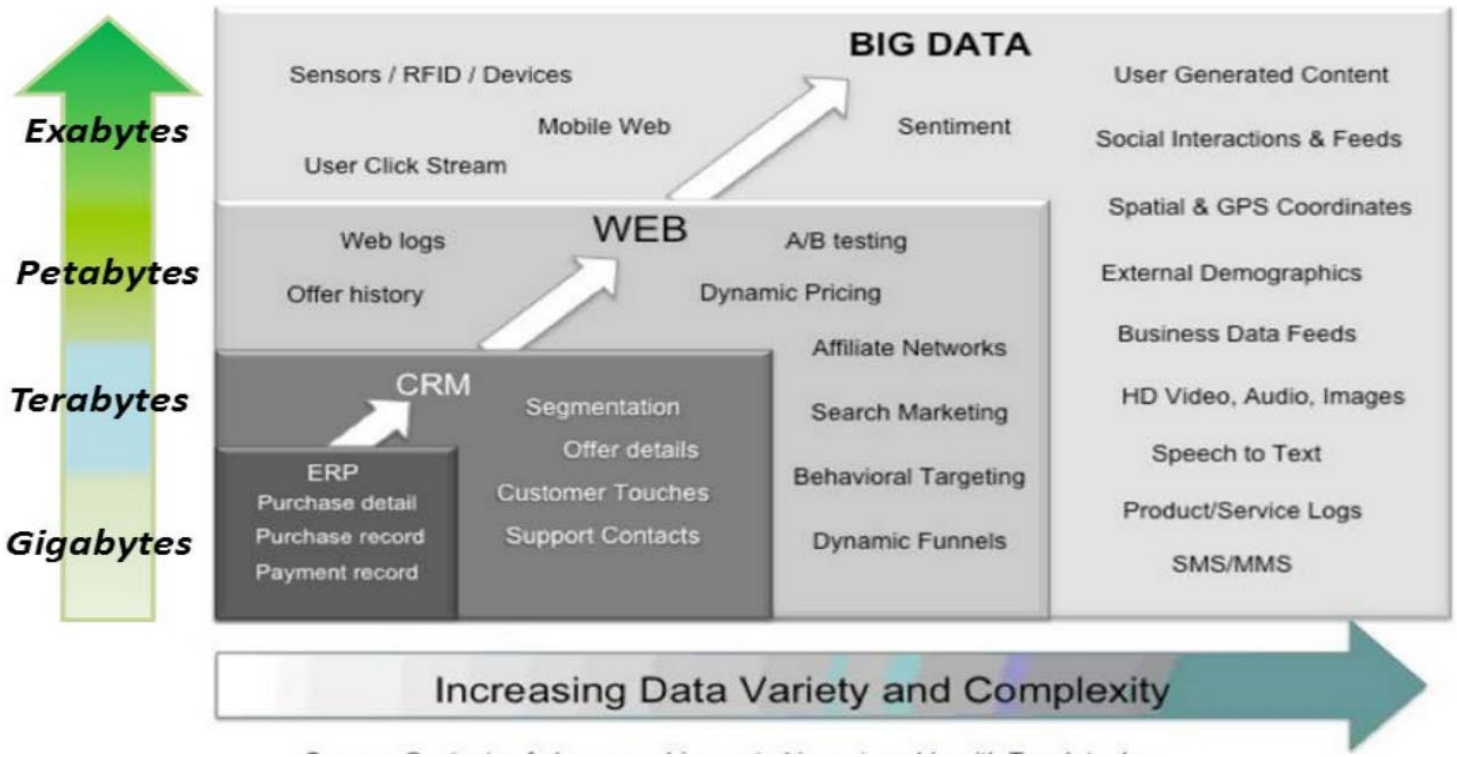
ARTEN VON DATEN

Hochschule RheinMain

- mobile Endgeräte (Smartphones, Watches, …): Multimedia-Daten (Fotos, Videos), Bewegungsdaten, Nachrichten …
- Unternehmensdaten – strukturierte Daten (Datenbanken, Data Warehouses) mit Angaben zu Personal, Kunden, Bestellungen, Rechnungen … – Dokumente, E-Mails
- Web-Daten – Web-Seiten + Multimedia-Inhalte (Videos, Fotos, …) – Click Logs
- soziale Netzwerke / Kommunikationsplattformen (Facebook, Twitter, Linkedin, ...) – Nutzer, Beziehungen, Nachrichten, Multimedia-Inhalte
- wissenschaftliche Daten zB in Klimaforschung, Experimentalphysik, Lebenswissenschaften und Sozialwissenschaften (Digital Humanities)
- Sensor-Daten / eingebettete Systeme – vernetzte Produktion/Fertigung (Industrie 4.0) – „intelligentes“ Haus, Verkehrsüberwachung, …
- IoT

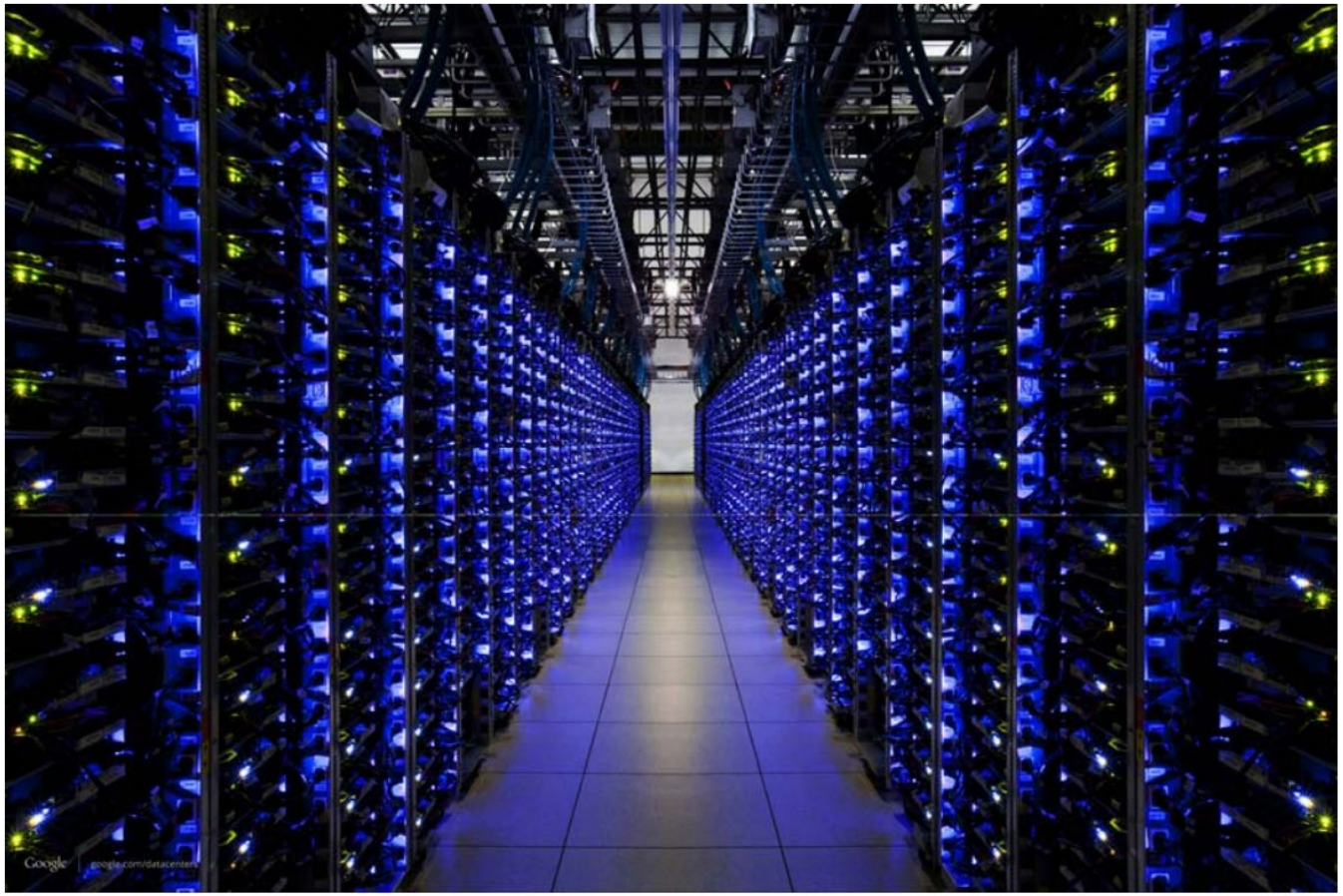
DATENZUNAHME

Hochschule RheinMain



DATA CENTER

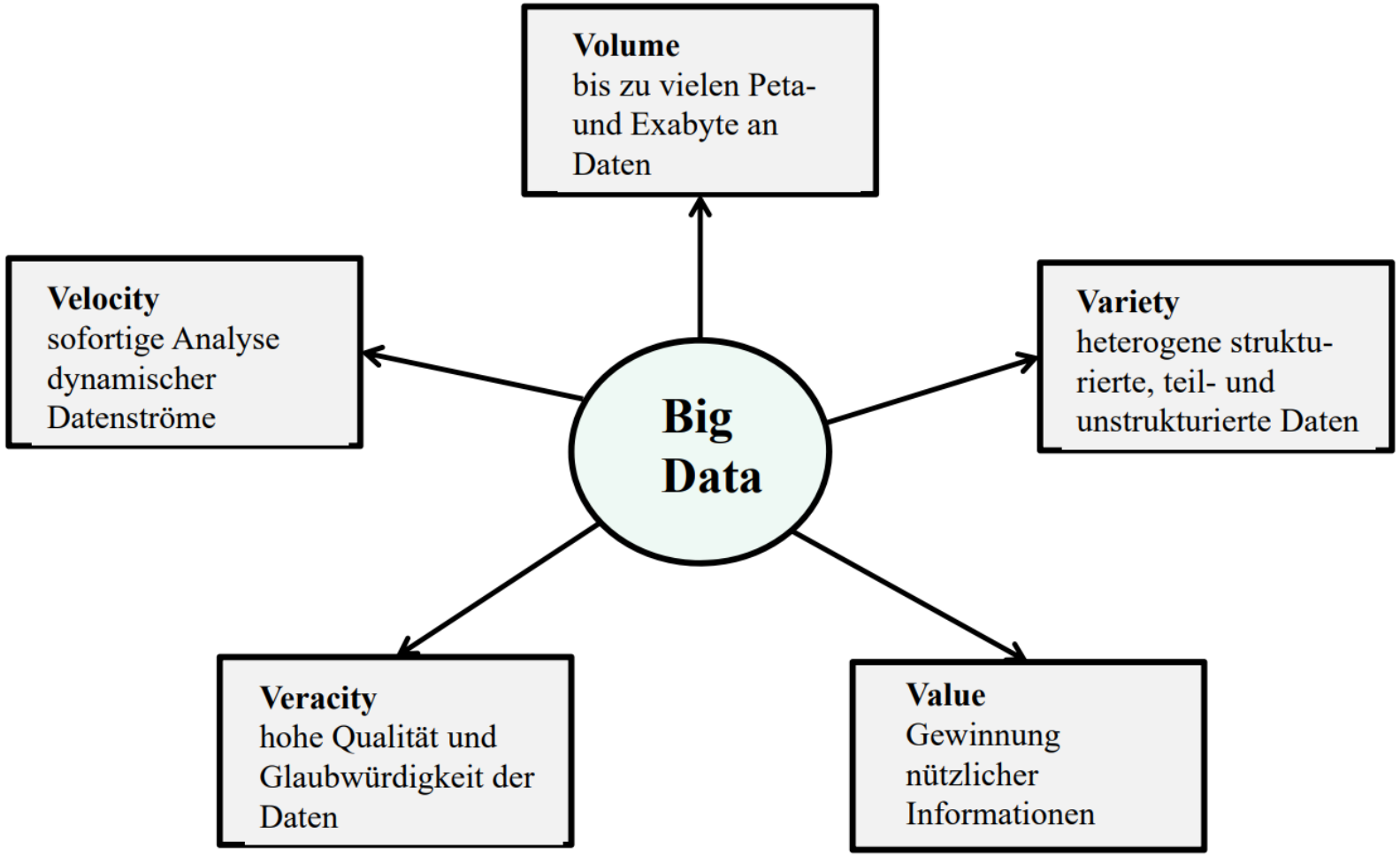
Hochschule RheinMain



Source: Google Inc.

BIG DATA CHALLENGES

Hochschule RheinMain



INDUSTRIE 4.0

Hochschule RheinMain

- Daten sind Produktionsfaktor: „data is the new oil“ – ähnlich Betriebsmitteln und Beschäftigten – essenziell für viele Branchen und Wissenschaftsbereiche
- valide Grundlage für zahlreiche Entscheidungsprozesse – Vorhersage/Bewertung/Kausalität von Ereignissen
- kurzfristige Analysen von Realdaten im Geschäftsleben – Empfehlungsdienste (Live Recommendations) – Analyse/Optimierung von Produktionsprozessen, Lieferketten, etc.
- verbesserte Analysen in zahlreichen wissenschaftlichen Anwendungen

ANWENDUNGEN

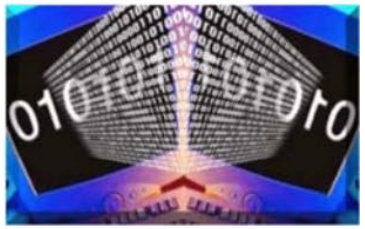
Hochschule RheinMain

Smarter Healthcare

Multi-channel sales

Finance

Log Analysis



Homeland Security

Traffic Control

Telecom

Search Quality

Manufacturing

Trading Analytics

Fraud and Risk

Retail: Churn, NBO

MAP REDUCE

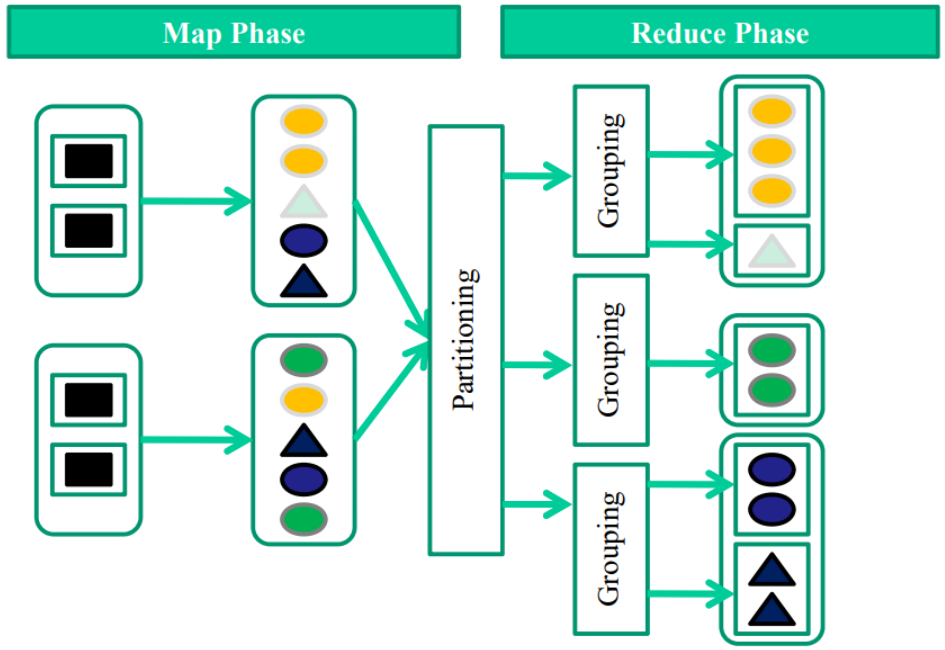
Hochschule RheinMain

- Framework zur automatischen Parallelisierung von Auswertungen auf großen Datenmengen
  - Entwicklung bei Google
  - populäre Open-Source-Implementierung: Hadoop
- Nutzung v.a. zur Verarbeitung riesiger Mengen teilstrukturierter Daten in einem verteilten Dateisystem
  - Konstruktion Suchmaschinenindex
  - Clusterung von News-Artikeln
  - Spam-Erkennung …

MAP REDUCE

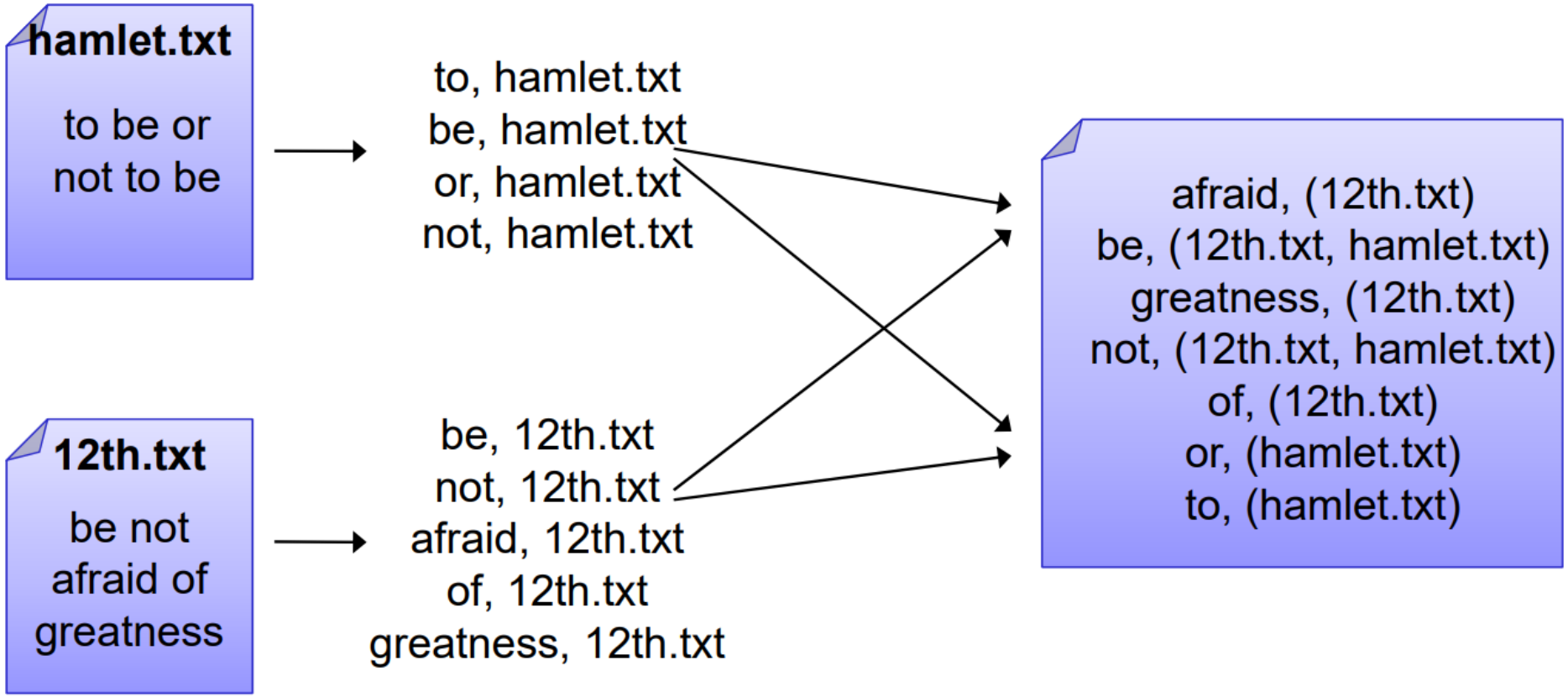
Hochschule RheinMain

- Verwendung zweier Funktionen: **Map** und **Reduce**
- **Map**-Anwendung pro Eingabeobjekt zur Erzeugung von Key-value Paaren
  - jedes Key-Value-Paar wird einem Reduce-Task zugeordnet
- **Reduce**-Anwendung für jede Objektgruppe mit gleichem Key



MR-BEISPIEL: GENERIERUNG TEXT-INDEX

Hochschule RheinMain



Hochschule RheinMain

THEORETISCHE GRUNDLAGEN

Map/Reduce, CAP

MAP/REDUCE-FRAMEWORK

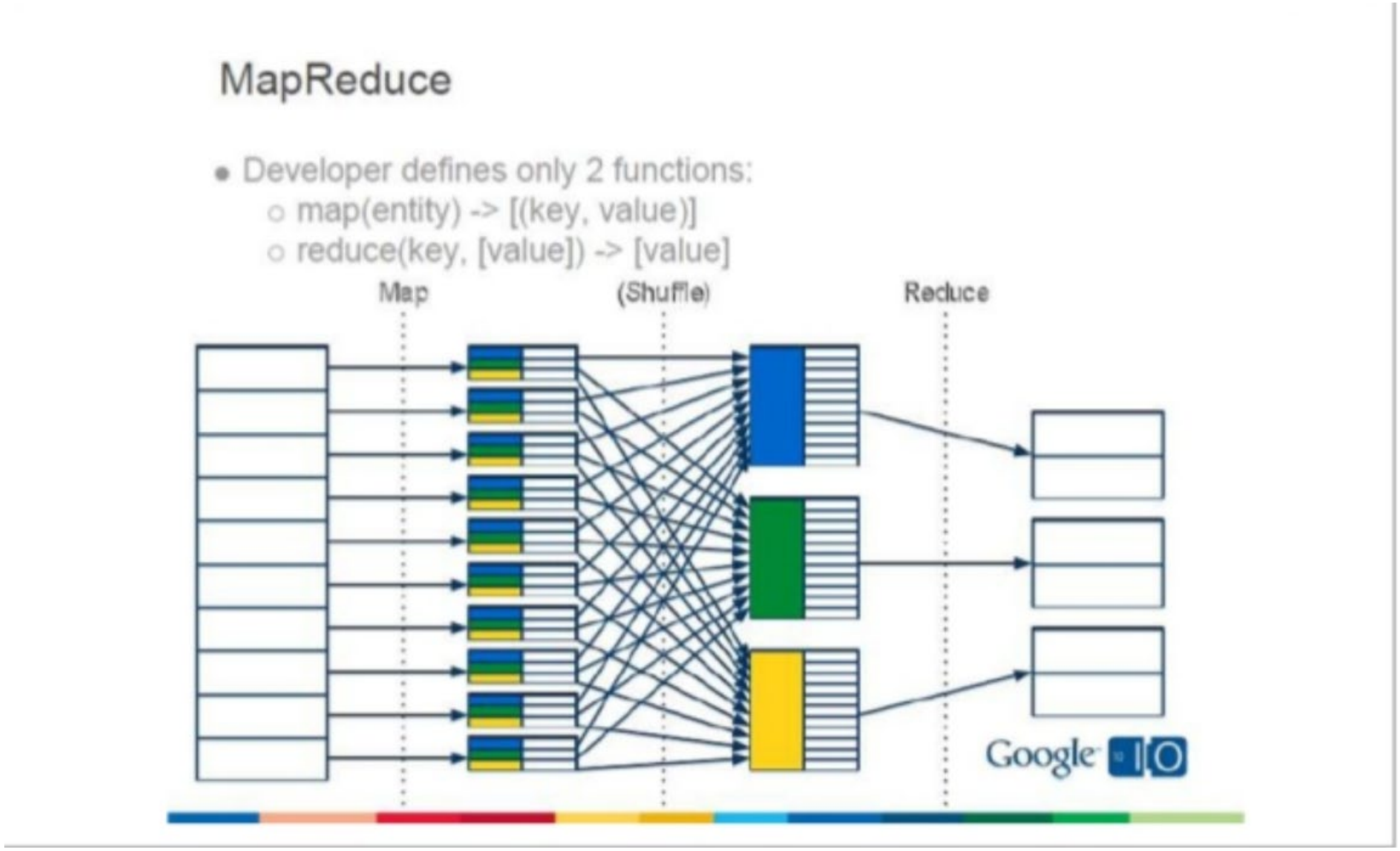
Hochschule RheinMain

- Entwickelt wurde das Map/Reduce-Framework 2004 bei Google Inc. von den Entwicklern Jeffrey Dean und Sanjay Ghemawat.
- Eine erste Vorstellung und Demonstration erfolgte auf der Konferenz „OSDI 04, Sixth Symposium on Operating System Design and Implementation“ in San Francisco, Kalifornien
- Im Januar 2010 hat Google Inc. auf das dort vorgestellte Map/Reduce-Verfahren vom US amerikanischen Patentbüro ein Patent erhalten.

- Die grundlegende Idee, Komponenten, Architektur und Anwendungsbereiche des Map/Reduce-Verfahrens werden gleich vorgestellt und beschrieben
- Die zahlreichen Implementierungen, die seit der ersten Vorstellung dieses Verfahrens entwickelt wurden, werden zur Übersicht kurz aufgelistet.

MAP REDUCE

Hochschule RheinMain



FUNKTIONALE URSPRUNGE

Hochschule RheinMain

- Die Parallelisierung von Prozessen beginnt bei der Formulierung von Algorithmen.
- Parallelisierung ist eine Stärke der funktionalen Sprachen.
- Die Grundidee von Map/Reduce kommt daher auch von funktionalen Programmiersprachen wie LISP3 und ML4.
- Vorteil funktionale Sprachen: keine Seiteneffekte wie Verklemmungen (*deadlock*) und Wettlaufsituationen (*race conditions*).
- Keine Veränderung vorhandenen Datenstrukturen
- arbeiten immer auf neu erstellten Kopien vorhandener Daten
- Die Originaldaten bleiben unverändert erhalten
- Unterschiedliche Operationen auf dem gleichen Datensatz beeinflussen sich somit nicht gegenseitig, da jede Operation auf einer eigenen Kopie der Originaldaten angewendet wird oder bei Datenergänzungen eine neue Datenstruktur erzeugt wird.
- Ausführungsreihenfolge von Operationen keine Rolle, wodurch die Parallelisierung dieser Operationen möglich wird.

FUNKTIONALE URSPRUNGE

Hochschule RheinMain

- Eine Funktion besteht dann aus einer Reihe von Definitionen, die diese Vorschrift beschreibt.
- Ein funktionales Programm besteht ausschließlich aus Funktionsdefinitionen und besitzt keine Kontrollstrukturen wie Schleifen.
- Wichtigstes Hilfsmittel für die funktionale Programmierung ist daher die Rekursion.
- Funktionen sind in funktionalen Programmiersprachen Objekte, mit denen wie mit Variablen gearbeitet werden kann. Insbesondere können Funktionen als Argument oder Rückgabewert einer anderen Funktion auftreten. Man spricht dann von Funktionen höherer Ordnung.

FUNKTIONALE URSPRUNGE

Hochschule RheinMain

- Die aus der funktionalen Programmierung bekannten Routinen *map()* und *fold(),* auch als *reduce()* bezeichnet, werden in modifizierter Form im Map/Reduce-Algorithmus jeweils nebenlaufig in zwei Phasen ausgeführt.
- Sie zahlen zu den Funktionen höherer Ordnung.

- Wie der Name der ältesten funktionalen Programmiersprache LISP (= **Lis**t **P**rocessing) schon verrat, geht es dabei um die Verarbeitung von Listen.
- Die Funktion *map()* wendet eine Funktion sukzessive auf alle Elemente einer Liste an und gibt eine durch die Funktion modifizierte Liste zurück.
- Die Funktion *reduce()* akkumuliert einzelne Funktionsergebnisse der Listenpaare und reduziert sie damit auf einen Ausgabewert.
- Diese beiden Funktionen werden in modifizierter Ausprägung als Map/Reduce-Algorithmus jeweils parallel auf verschiedenen Knoten im Netzwerk in zwei Phasen hintereinander angewendet.

MAP FUNKTION

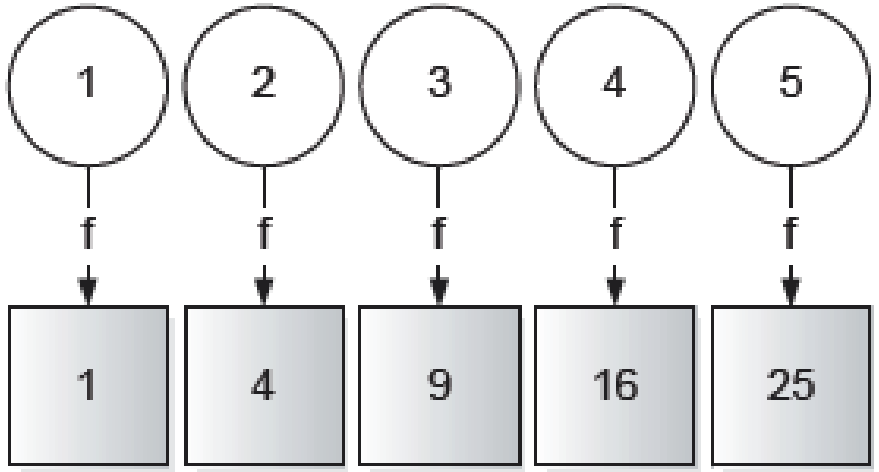
Hochschule RheinMain

- Die *map()*-Funktion der funktionalen Sprachen erhält als Argument eine Funktion f und wendet diese auf jedes Element einer übergebenen Liste an. Es ist eine polymorphe Funktion, die beliebige Argumenttypen erhalten kann, wie durch die Typvariablen a und b angegeben ist.

```
map :: (a -> b) -> [a] -> [b]
map f []          = []
map f (x : xs)    = f x : map f xs
```

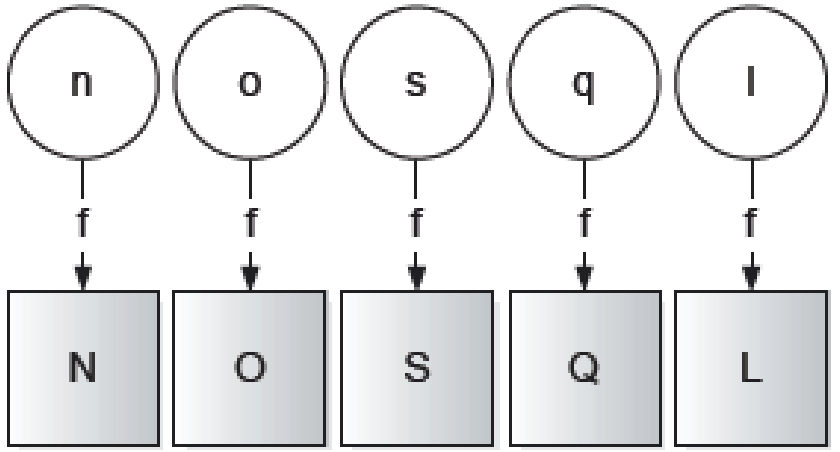
```
map (\x -> x^2) [1,2,3,4,5]
```

```
[1,4,9,16,25]
```



```
:module +Data.Char
map toUpper "nosql"
```

```
"NOSQL"
```



FOLD FUNKTION

Hochschule RheinMain

- Die *fold()*-Funktion realisiert quasi eine *n*-zu-1-Transformation und wird in diesem Zusammenhang auch in anderen Sprachen als *reduce-, accumulate-*, *compress-* oder *inject-*Funktion bezeichnet.

- Das Ergebnis dieser Transformation muss nicht aus einem Element bestehen, es kann auch wiederum eine Liste von reduzierten Elementen sein.

- In Haskell wie auch in vielen anderen funktionalen Programmiersprachen werden zwei Varianten von *fold()* unterschieden:
  - die *foldl*-Funktion für die Bearbeitung einer Liste von links nach rechts und
  - die *foldr*-Funktion für die Bearbeitung einer Liste von rechts nach links

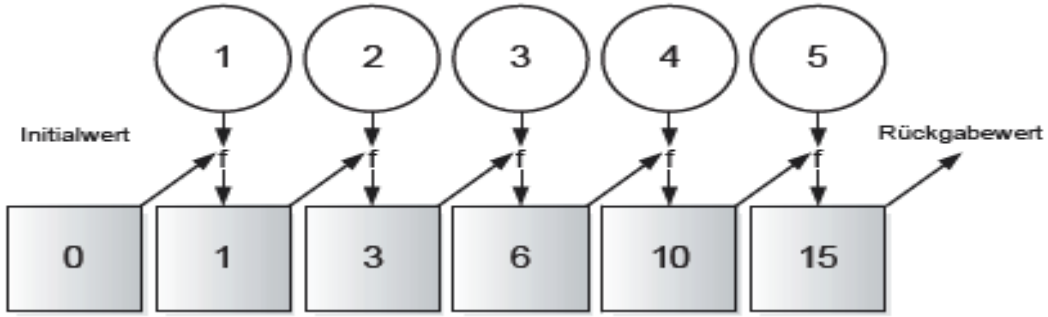
```
foldl :: (a -> b -> a) -> a -> [b] -> a
foldl f z []     = z
foldl f z (x:xs) = foldl f (f z x) xs
```

```
foldr :: (a -> b -> b) -> b -> [a] -> b
foldr f z []     = z
foldr f z (x:xs) = f x (foldr f z xs)
```

```
foldl (+) 0 [1,2,3,4,5,6,6,7,8]
foldr (+) 0 [1,2,3,4,5,6,6,7,8]
```

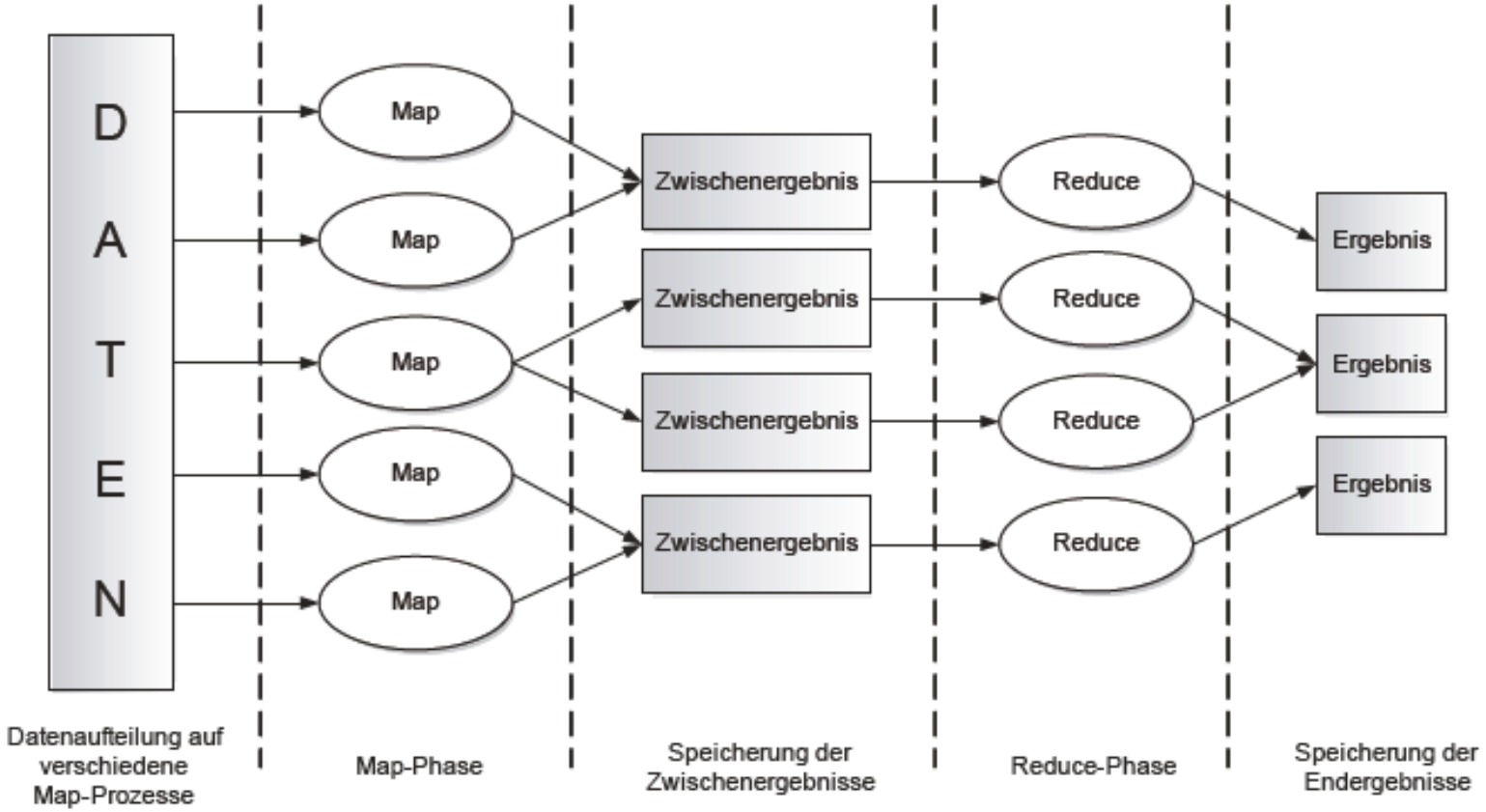
Eine Berechnung mit foldl entspricht: (((((((0+1) + 2) + 3) + 4)+5)+6)+6)+7)+8

③ Eine Berechnung mit foldr entspricht: 1 + (2 + (3 + (4 + (5+(6+(6+(7+(8+0)))))))))



PHASEN UND DATENFLUSS

Hochschule RheinMain



- Zuerst werden alle Eingabedaten auf verschiedene Map-Prozesse aufgeteilt.
  - In der Map-Phase berechnen diese Prozesse jeweils parallel die vom Nutzer bereitgestellte Map-Funktion.
  - Von jedem Map-Prozess fliesen Daten in verschiedene Zwischenergebnisspeicher.
  - Die Map-Phase ist beendet, sobald alle Zwischenergebnisse berechnet worden sind.
- Nun beginnt die Reduce-Phase, in der für jeden Satz an Zwischenergebnissen ein Reduce-Prozess die vom Nutzer bereitgestellte Reduce-Funktion parallel berechnet.
  - Jeder dieser Reduce-Prozesse speichert seine Ergebnisdatei im Dateisystem ab.
  - Die Reduce-Phase ist beendet, sobald alle Ergebnisdateien abgespeichert worden sind.
- Somit ist auch die gesamte Durchführung des Map/Reduce-Verfahrens beendet.

- Map-Funktion und Reduce-Funktion muss vom Anwender erstellt werden.
- die Logik seiner Anwendung
- Die Verteilung der Daten, der Map-Prozesse, der Reduce-Prozesse sowie die Speicherung der Zwischen-und Endergebnisse werden im Wesentlichen vom Map/Reduce-Framework übernommen.
- Die Trennung zwischen Anwendungslogik und technischer Seite des Frameworks ermöglicht es dem Anwender, sich auf die Losung seines Problems zu konzentrieren.
- Das Framework übernimmt unterdessen intern die schwierigen Details der
  - automatischen Parallelisierung und Verteilung der Prozesse,
  - Realisierung von Fehlertoleranz bei Ausfall von Hardware und Software,
  - I/O-Scheduling,
  - Bereitstellung von Statusinformationen und Uberwachungsmoglichkeiten.
- Der Programmierer muss lediglich die beiden namensgebenden Funktionen *map()* und *reduce()* spezifizieren, welche nachfolgend im Pseudocode angegeben werden.

```
map (in_key, in_value) -> list(out_key, intermediate_value)
reduce (out_key, list(intermediate_value)) -> list(out_value)
```

ANALYSE DER WORTHÄUFIGKEIT IN EINEM UMFANGREICHEN TEXT.

Hochschule RheinMain

```
map(String key, String value):
// key: document name
// value: document contents
for each word w in value:
 EmitIntermediate(w, "1");
```

```
reduce(String key, Iterator values):
// key: a word
// values: a list of counts
int result = 0;
for each v in values:
 result += ParseInt(v);
Emit(AsString(result));
```

ANWENDUNGSBEREICHE UND IMPLEMENTIERUNGEN

Hochschule RheinMain

- **Verteiltes Suchen (Grep)**: Beim Suchen von Mustern über verteilte Daten liefert die Map-Funktion eine Zeile, wenn der Inhalt der Zeile dem Suchmuster entspricht. Die Reduce-Funktion ist eine Identitätsfunktion welche die Zwischenergebnisse nur zur Liste der Endergebnisse kopiert.

- **Zählen von Zugriffen auf eine URL**: Hier verarbeiten die Map-Funktionen die Zugriffe auf Webseiten aus vorhandenen Aufzeichnungen und geben, ähnlich dem Zahlen der Worthäufigkeit, die Key/Value-Paare (URL,“1“) aus. Die Reduce-Funktionen addieren diese Zwischenergebnisse pro URL und liefern als Ergebnis die Zugriffe pro URL (URL,total_count).

- **Erstellung von Graphen über Verlinkung von Webseiten zu einem Ziel**: Die Map-Funktionen erzeugen Key/Value-Paare in der Form (target,source) von jeder Verbindung zu einer Ziel-URL (target), die auf einer Quellen-URL (source) gefunden wird. Die Reduce-Funktionen konkatenieren die Liste aller Quellen-URLs in Verbindung mit der Ziel-URL und liefern diese Verbindungen als Ergebnis in der Form von (target,list(source)).

WEITERE ANWENDUNGSBEREICHE

Hochschule RheinMain

- **Ermittlung des Term-Vectors per Host**: Ein *Term-Vector* fasst die wichtigsten Worte zusammen, die in einem Dokument oder in eine Gruppe von Dokumenten auftreten. Dieser Vector wird in Form einer Liste von Wort-Wortfrequenz-Paaren (word, frequency) dargestellt. Die Map-Funktionen liefern für jedes zum *Host* gehörende Dokument die Key/Value-Paare (hostname,term_vector). Die Reduce-Funktionen überprüfen alle Term-Vectoren pro Dokument und fassen diese Vectoren für die Dokumente zusammen. Dabei werden Terme mit geringer Frequenz verworfen und ein abschließendes Paar (hostname,term_vector) geliefert.
- **Wortindex Erstellung**: Die Map-Funktionen analysieren jedes Dokument und liefern Key/Value-Paare in Form von Wort-Dokumentennummer-Paaren (word, document_ID). Die Reduce-Funktionen sortieren diese Zwischenergebnisse für jedes Wort nach der Dokumentennummer und fassen diese Daten als Liste zusammen. Als Ergebnis wird ein invertierter Wordindex geliefert (word,list(document_ID).

ZUSAMMENFASSUNG

Hochschule RheinMain

- Das Map/Reduce-Verfahren spielt im Kontext der NoSQL-Datenbanken eine zentrale Rolle.
- Damit lassen sich große verteilte Datenmengen bei paralleler Ausführung effizient durchsuchen.
- NoSQL-Datenbanken wie z.B. CouchDB, MongoDB, Riak und HBase nutzen das Map/Reduce-Verfahren zur Abfrage ihrer Datenbankeintrage.
- Die Ursprünge des Map/Reduce-Verfahrens liegen in der funktionalen Programmierung, welches die Parallelisierung innerhalb der Map/Reduce-Phasen ermöglicht.
- Die zur Losung einer Aufgabe benötigte Map-Funktion und Reduce-Funktion muss vom Anwender erstellt werden. Mit ihnen definiert er die Logik seiner Anwendung. Die Verteilung der Daten, von Map- und Reduce-Prozessen sowie die Speicherung der Zwischen- und Endergebnisse werden im Wesentlichen vom Map/Reduce-Framework übernommen.
- Das Framework ermöglicht außerdem
  - Parallelisierung
  - Fehlertoleranz
  - Datentransfer
  - Lastverteilung
  - Monitoring