Non-Standard-Datenbanken

Von NoSQL zu NewSQL

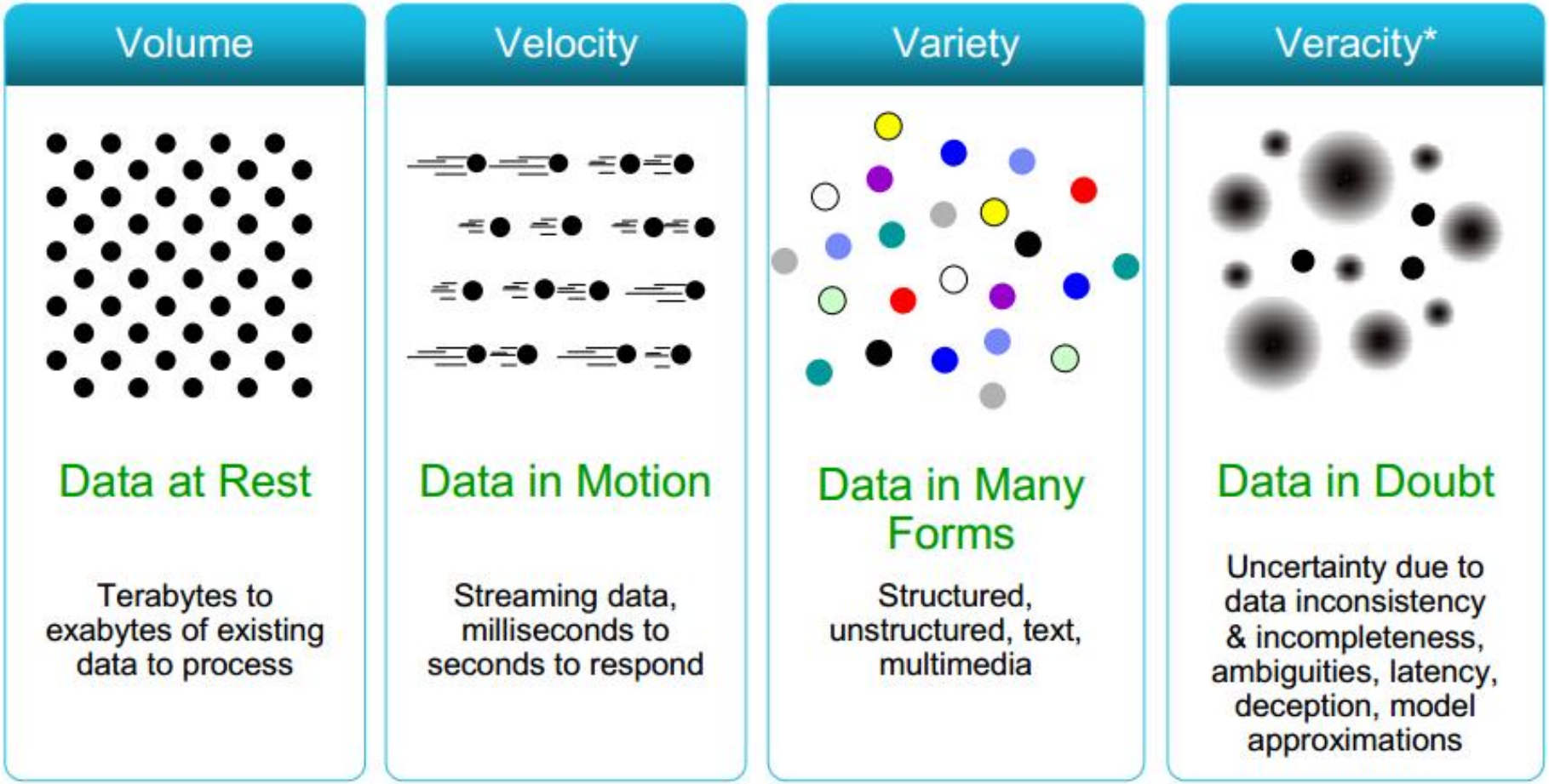
Prof. Dr. Ralf Möller

Universität zu Lübeck

Institut für Informationssysteme

Dennis Heinrich (Übungen)

Big Data

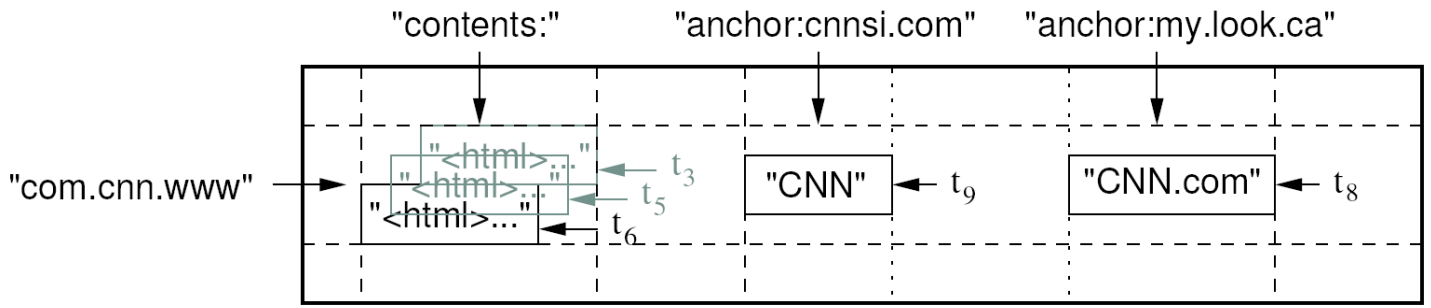

Volume
Data at Rest
Terabytes to exabytes of existing data to process
Velocity
Data in Motion
Streaming data, milliseconds to seconds to respond
Variety
Data in Many Forms
Structured, unstructured, text, multimedia
Veracity*
Data in Doubt
Uncertainty due to data inconsistency & incompleteness, ambiguities, latency, deception, model approximations

NoSQL: Not Only SQL?

- Datenmengen werden groß (Big Data)
  - Horizontale Skalierung notwendig
  - Virtualisierung (Cloud Computing) geht damit einher
- Heterogenität von Daten, Datenintegration
  - Schema-Freiheit (relationales Modell scheint zu starr)
  - Algorithmische Datenverarbeitung wird gewünscht
- Verteilung der Datenhaltung
  - CAP Theorem (Consistency, Availability, Paritioning: Eric Brewer)
    - Nur 2 aus 3 Kriterien erfüllbar
  - Abschwächung von des Konsistenzkriteriums: BASE
    - **B**asically **A**vailable
    - **S**oft-state (or scalable)
    - **E**ventually consistent

Bigtable (Google) und Dynamo (Amazon)

- Eine Tabelle in Bigtable ist eine dünn besetzte, multidimensionale, persistente und verteilte Hashtabelle
- Abbildung ist indexiert durch Zeilenschlüssel, Spaltenschlüssel und Zeitstempel
  - (row:string, column:string, time:int64) → uninterpretiertes Bytefeld
- Unterstützung von Suchen, Einfügen, Löschen
  - Transaktionen nur zeilenweise etablierbar



F. Chang, J. Dean, S. Ghemawat, W.C. Hsieh, D.A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R.E. Gruber. Bigtable: A Distributed Storage System for Structured Data. *ACM Trans. Comput. Syst.* 26, 2, Article 4, **2008**

Image Source: Chang et al., OSDI 2006

Map Reduce

- Technik zur Indizierung und Verarbeitung von großen Datenmengen
- Implementierung durch Anwendungsentwickler, kein deklaratives DBMS
- Zwei Phasen: Map und Reduce
  - Map
    - Extraktion von Mengen von Key-Value-Paaren aus Daten
    - Potentiell auf vielen Maschinen parallel
  - Reduce
    - Mischung und Sortierung von Key-Value-Paaren
    - Resultat in anderen Verarbeitungskontexten weiterverwendet

Map-Reduce-Patent

Google granted US Patent 7,650,331, January 2010

System and method for efficient large-scale data processing

A large-scale data processing system and method includes one or more application-independent map modules configured to read input data and to apply at least one **application-specific map operation** to the input data to produce intermediate data values, wherein the map operation is automatically parallelized across multiple processors in the parallel processing environment. A plurality of intermediate data structures are used to store the intermediate data values. One or more application-independent reduce modules are configured to retrieve the intermediate data values and to apply at least one **application-specific reduce operation** to the intermediate data values to provide output data.

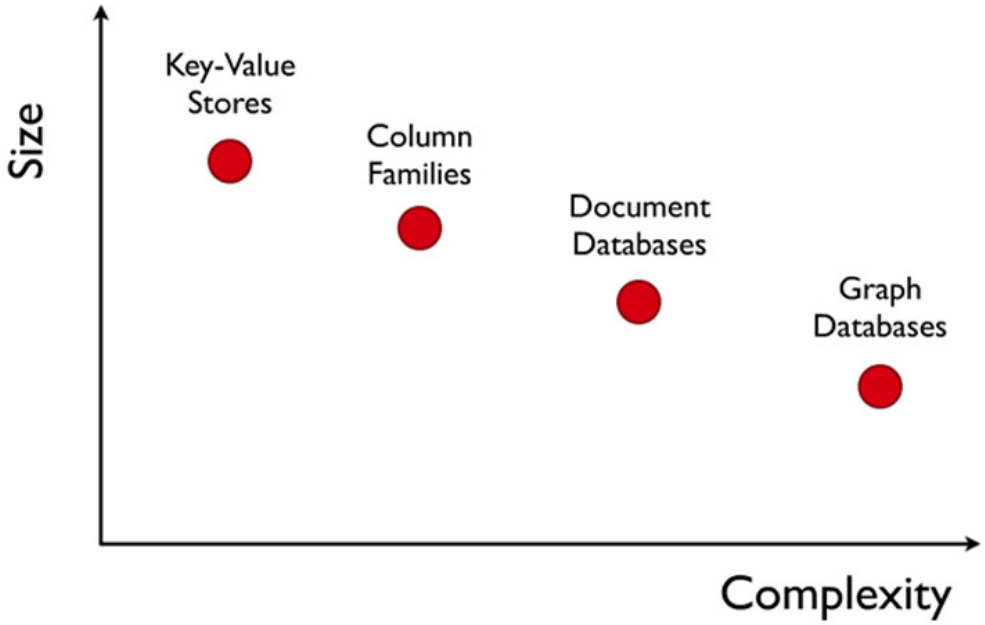
NoSQL-Beispiel: Key-Value Store

- Hash-Tabelle von Schlüsseln
- Werte mit Schlüsseln zusammen gespeichert
- Schneller Zugriff auf (kleine) Datenwerte
- Voldemort
  - http://www.project-voldemort.com/
- MemCacheDB
  - http://memcachedb.org/
  - Backend-Speicher ist eingebettete DB (Berkeley-DB)

NoSQL-Datenbanktypen

Die Diskussion von NoSQL-Datenbanken wird erschwert durch Vielzahl verschiedener Typen:

- Key-Value Store – Hashtabelle von Schlüsseln
- Column Store – Jeder Speicherblock enthält Daten aus nur einer Spalte
- Document Store
  "Dokumente" als Menge ausgezeichneter (tagged) Elemente
- Graphdatenbanken



Andere Nicht-SQL-Datenbanken

- Hierarchische Datenbanken (IBM, 60er Jahre)
- Netzwerk-Datenbanken (CODASYL, 70er Jahre)
- Objekt-orientierte Datenbanken (80er Jahre)
- XML-Datenbanken (90er Jahre)
- Triple Stores (2000er Jahre → Sven Groppes Vorlesung)
- …

NoSQL-Beispiel: Column Store

- Jeder Speicherblock enthält Daten aus nur einer Spalte
  - Effizienzgewinn, wenn nur einige Spalten relevant für Anfrage
- MonetDB
  - https://www.monetdb.org
  - MonetDB (seit 1993)
- Hadoop/Hbase
  - http://hadoop.apache.org/
  - Yahoo, Facebook
- Actian Vortex (ex: VectorWise, ex: Vector)
  - Column Store integriert in SQL-Datenbank Ingres
  - http://www.actian.com
  - Vertreibt auch Versant, ein objektorientiertes DBMS

NoSQL-Beispiel: Document Store

- Verwendung von JSON – JavaScript Object Notation
  - Geschachtelte Objekte
  - Wie XML aber "bequemer" für Java-Programmierer
  - Aber: Keine Vermeidung von Redundanz
- CouchDB
  - http://couchdb.apache.org/
- MongoDB
  - http://www.mongodb.org/

CouchDB JSON Beispiel

```
{
  "_id": "guid goes here",
  "_rev": "314159",

  "type": "abstract",

  "author": "Keith W. Hare"

  "title": "SQL Standard and NoSQL Databases",

  "body": "NoSQL databases (either no-SQL or Not Only SQL)
           are currently a hot topic in some parts of
           computing.",
  "creation_timestamp": "2011/05/10 13:30:00 +0004"
}
```

CouchDB JSON Tags

- "_id"
  - GUID – Global Unique Identifier
  - Passed in or generated by CouchDB
- "_rev"
  - Revision number
  - Versioning mechanism
- "type", "author", "title", etc.
  - Arbitrary tags
  - Schema-less
  - Could be validated after the fact by user-written routine

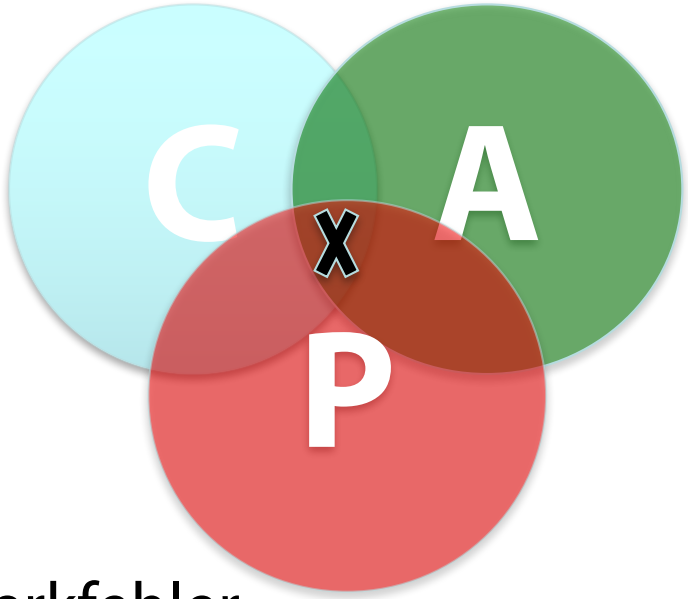
NoSQL: Zusammenfassung

- Nutzer von NoSQL-Datenbanken lehnen ab:
  - Aufwand von ACID-Transaktionen
  - “Komplexität” von SQL
  - Aufwand des Designs von Schemata
  - Deklarative Anfrageformulierung
  - Ein-Prozessor-Technologie
- Programmierer wird verantwortlich für
  - Prozedurale Formulierung von Zugriffsalgorithmen
  - und Navigation zu den benötigten Daten

- Jüngst: Revival von SQL in allen Systemen

CAP Theorem

- **C**onsistency:
  - Alle Knoten sehen die gleichen Daten zur gleichen Zeit
- **A**vailability (Verfügbarkeit):
  - Fehler von Knoten beeinträchtigen nicht die Funktionsfähigkeit der Überlebenden
- **P**artitionierungstoleranz:
  - System arbeitet weiter, auch bei Partitionierung durch Netzwerkfehler

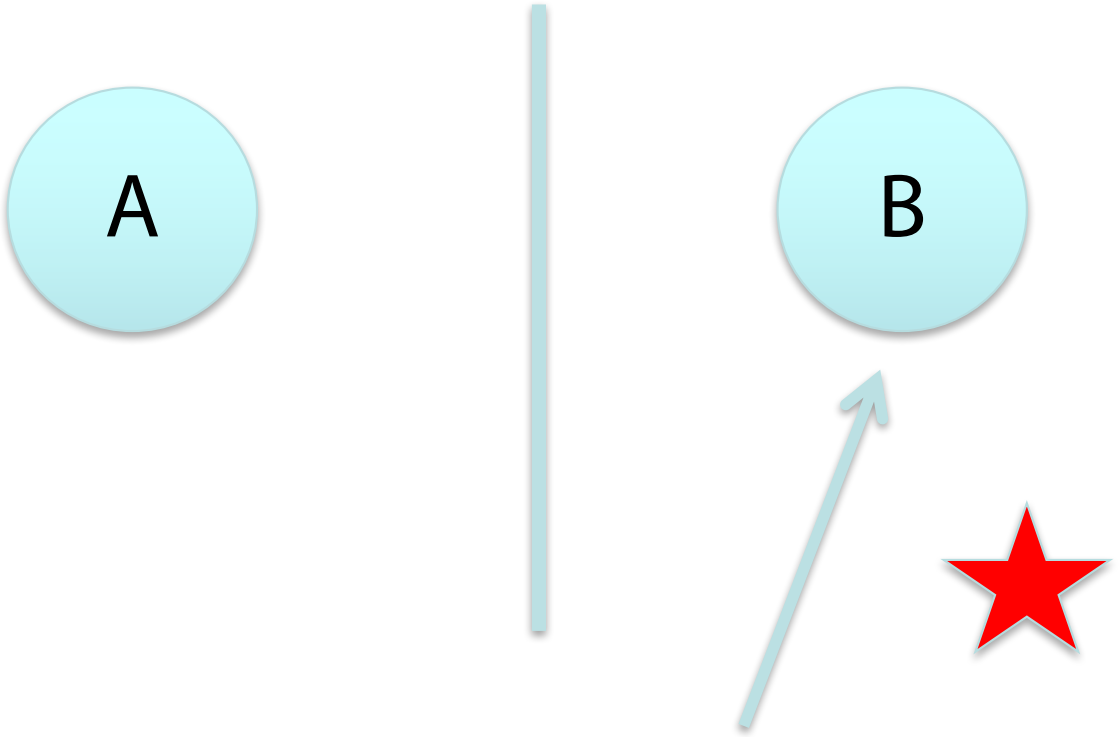


- Theorem: Ein verteiltes System kann nur jeweils zwei Kriterien erfüllen, nicht aber alle drei

Gilbert, Seth, and Nancy Lynch. "Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services." ACM SIGACT News 33.2 (**2002**): 51-59

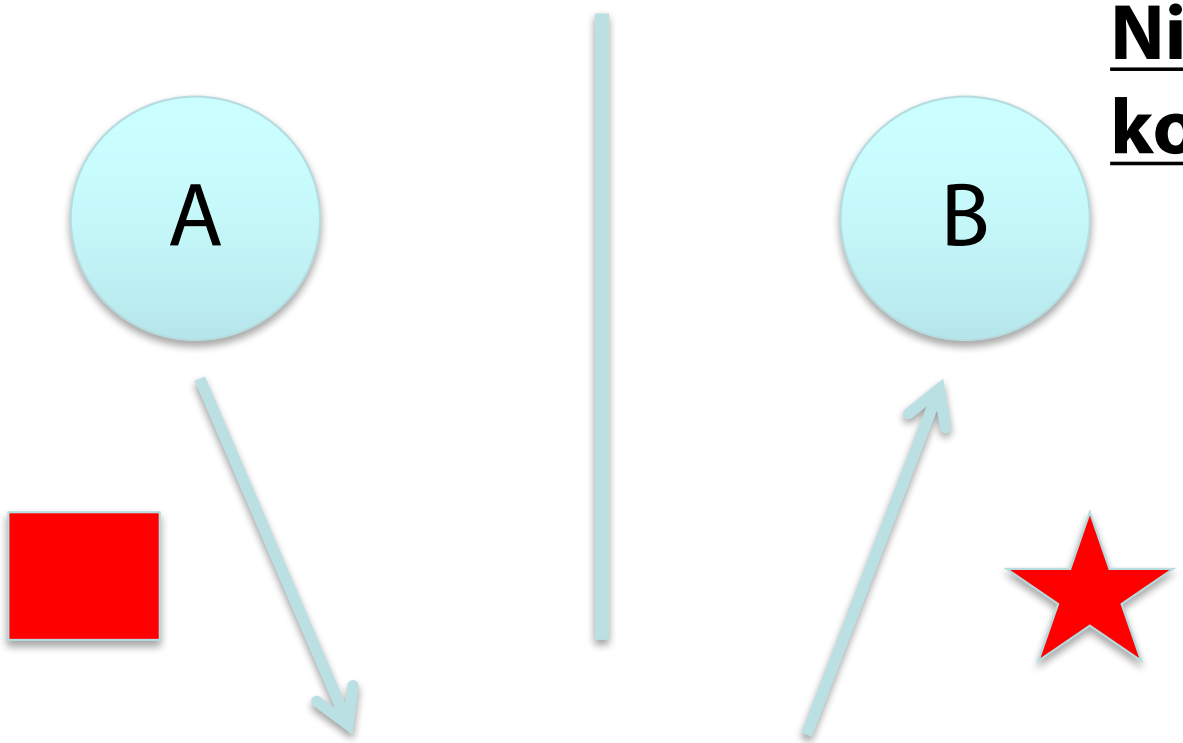
CAP Theorem: Argumente

- Nehmen wir an, es gibt zwei Knoten:



CAP Theorem: Argumente

- Nehmen wir an, es gibt zwei Knoten:



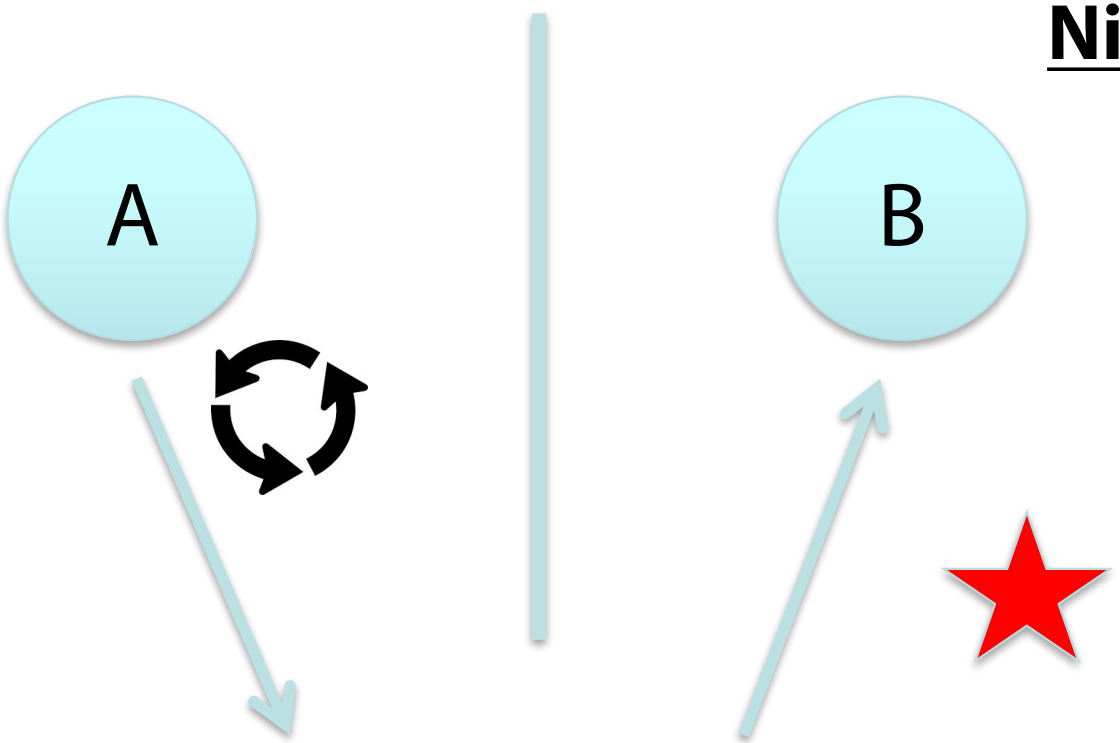
Nicht konsistent!

Antwort an Klienten

CAP Theorem: Argumente

- Nehmen wir an, es gibt zwei Knoten:

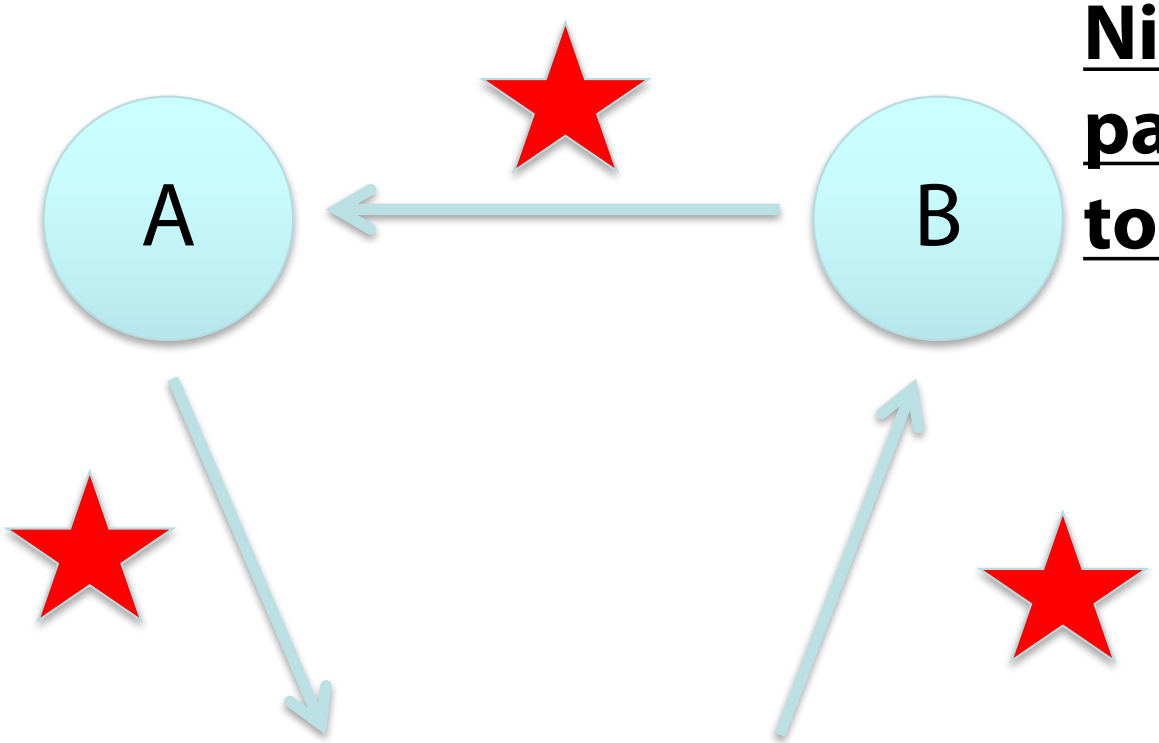
Nicht verfügbar!



Warten auf Aktualisierung

CAP Theorem: Argumente

- Nehmen wir an, es gibt zwei Knoten:



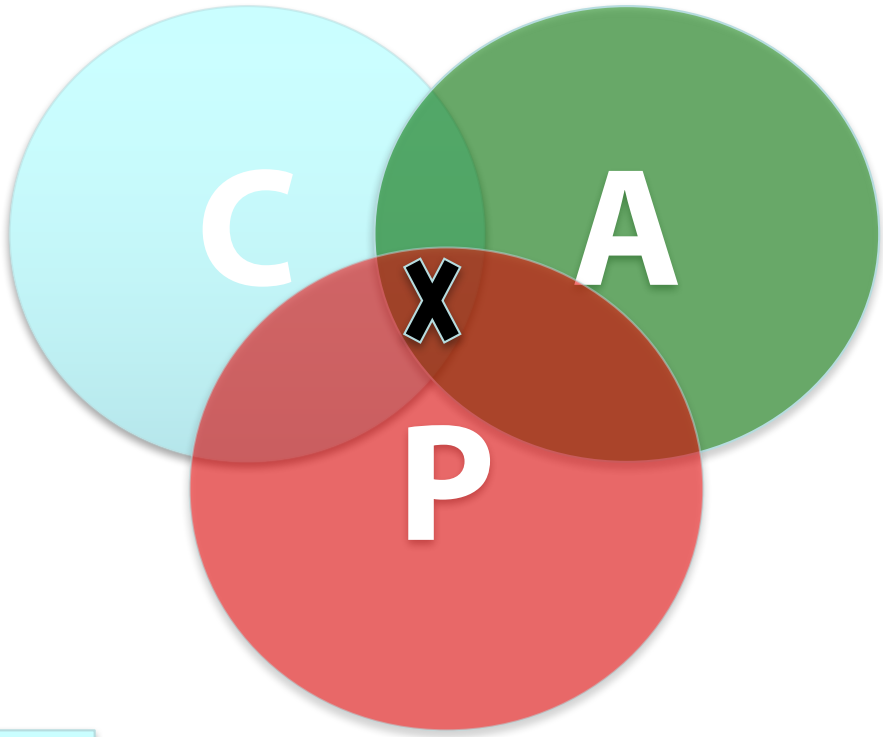
Nicht partitionierungs-tolerant!

A erhält Aktualisierung von B

Partitionierung:
System zerfällt in (mindestens) zwei unverbundenen Teile

Neubetrachtung CAP Theorem

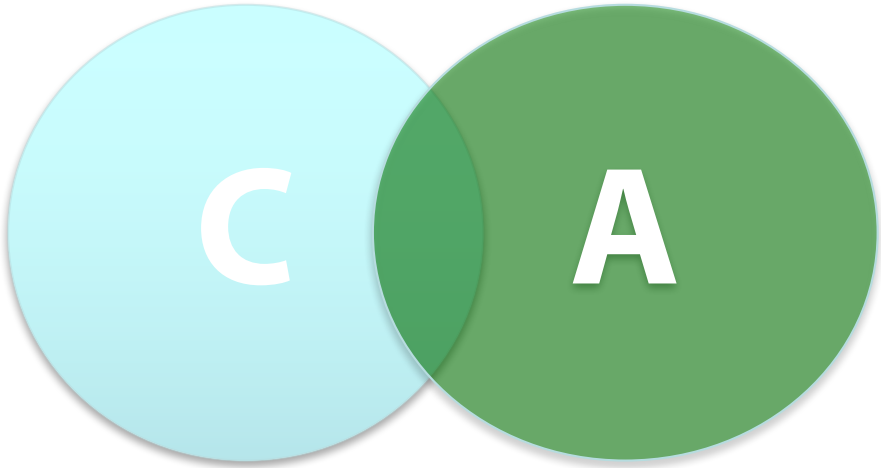
- Von den folgenden Garantien angeboten von verteilten Systemen:
  - Consistency
  - Availability
  - Partition tolerance
- Wähle zwei
- Drei Möglichkeiten :
  - CP
  - AP
  - CA



Probleme?

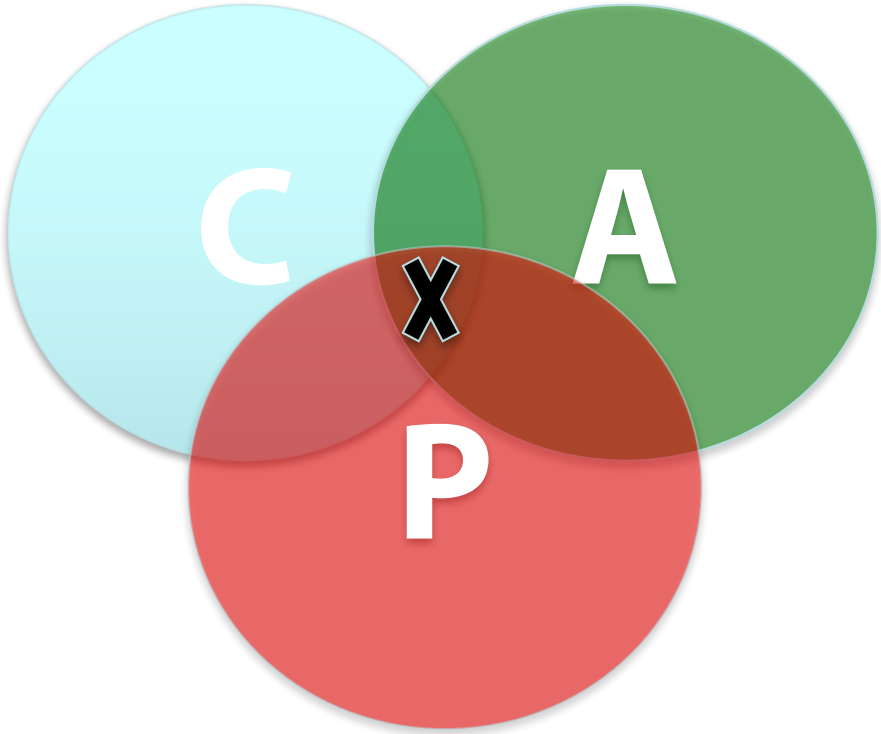
Häufiges Missverständnis: 2 von 3

- Was ist mit CA?
- Kann ein verteiltes System (mit unzuverlässigem Netzwerk) wirklich nicht partitionierungstolerant sein?



Konsistenz oder Verfügbarkeit

- Konsistenz und Verfügbarkeit sind keine binären Entscheidungen
- AP-Systeme schwächen Konsistenz zugunsten von Verfügbarkeit (sind aber nicht notwendigerweise inkonsistent)
- CP-Systeme opfern Verfügbarkeit zugunsten der Konsistenz (sind aber durchaus meist verfügbar)
- Quintessenz: AP- und CP-Systeme können Konsistenz, Verfügbarkeit und Partitionierungstoleranz graduell zur Verfügung stellen



Arten der Konsistenz

- Starke Konsistenz
  - Nach einer Datenaktualisierung muss jeder nachfolgende Zugriff (egal auf welche Kopie) den neuen Wert liefern
- Schwache Konsistenz
  - Es wird nicht garantiert, dass nachfolgende Zugriffe auf Kopien den aktualisierten Wert liefern
- LetztEndliche Konsistenz (eventual consistency)
  - Spezielle Form der schwachen Konsistenz
  - Es wir garantiert, dass schließlich alle Prozesse den letztmalig aktualisierten Wert sehen, wenn keine neuen Änderungen erfolgen (verzögert Propagierung von Änderungen in die Kopien)

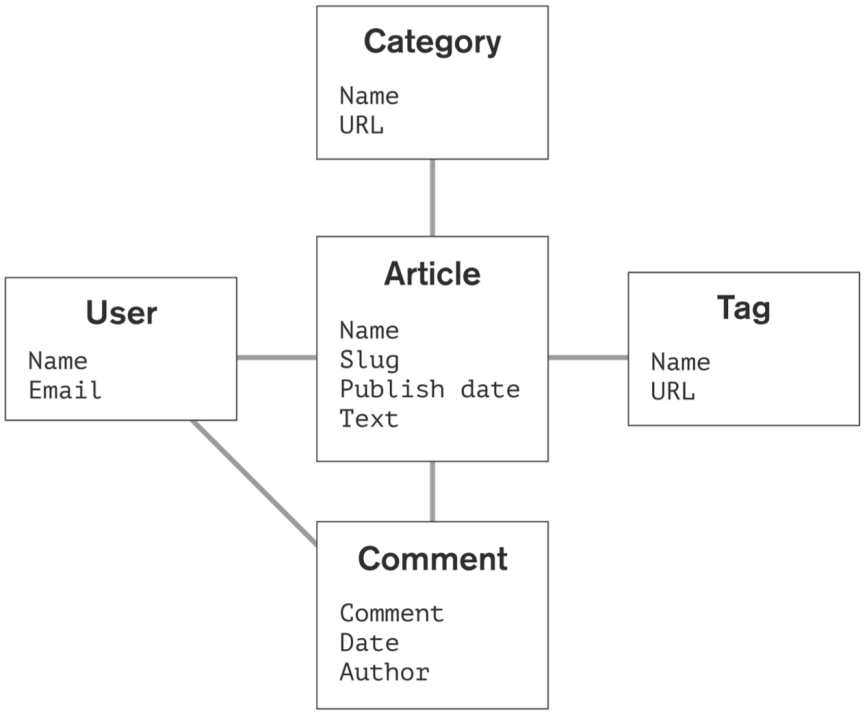
Abadi, Daniel J. "Consistency tradeoffs in modern distributed database system design." Computer-IEEE Computer Magazine 45.2 (**2012**): 37.

Beispiele

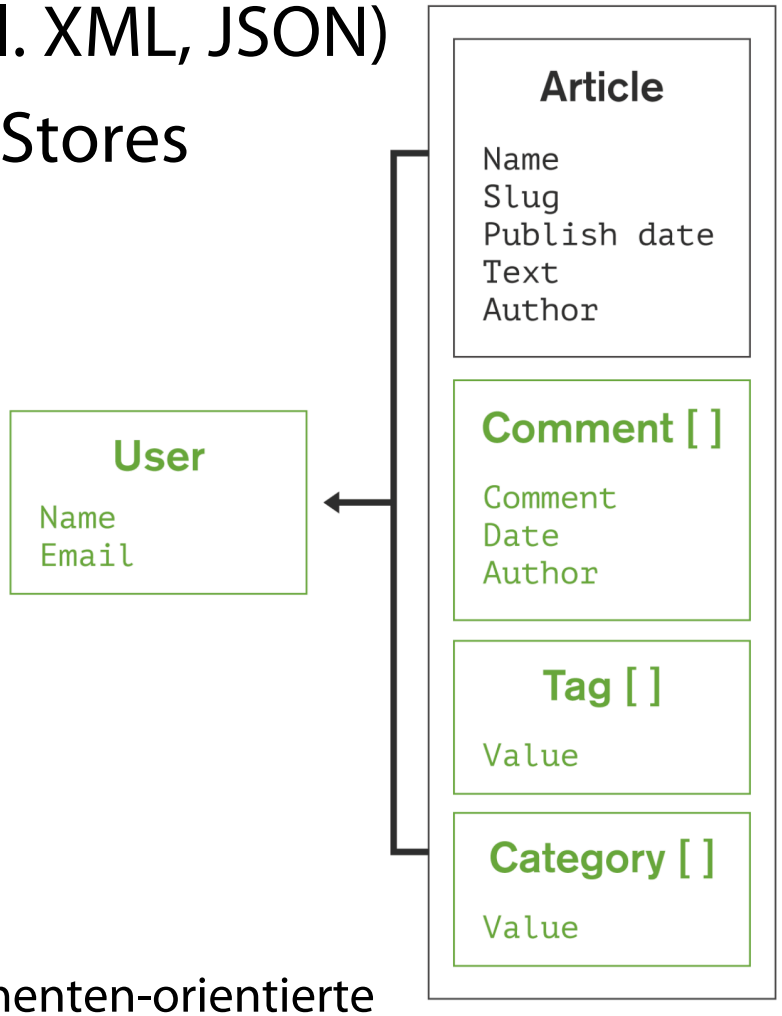
- **PA/EL-System:** Gib Konsistenz C auf zugunsten der Verfügbarkeit A und kleinerer Latenz L
  - Dynamo, Cassandra, Riak
- **PC/EC-System:** Verweigere, die Konsistenz C aufzugeben, und zahle die Kosten von Verfügbarkeit und Latenz
  - BigTable, Hbase, VoltDB/H-Store
- **PA/EC Systems:** Gib Konsistenz auf, wenn Partitionierung erfolgt, und erhalte Konsistenz im normalen Betrieb
  - MongoDB
- **PC/EL System:** Erhalte Konsistenz, wenn Partitionierung erfolgt, gebe Konsistenz für kleine Latenz im normalen Betrieb auf
  - Yahoo! PNUTS

Dokument-orientierte DB: Beispiel MongoDB

- Daten als "Dokumente" (vgl. XML, JSON)
- Unterklasse von Key-Value-Stores



Relationale Darstellung



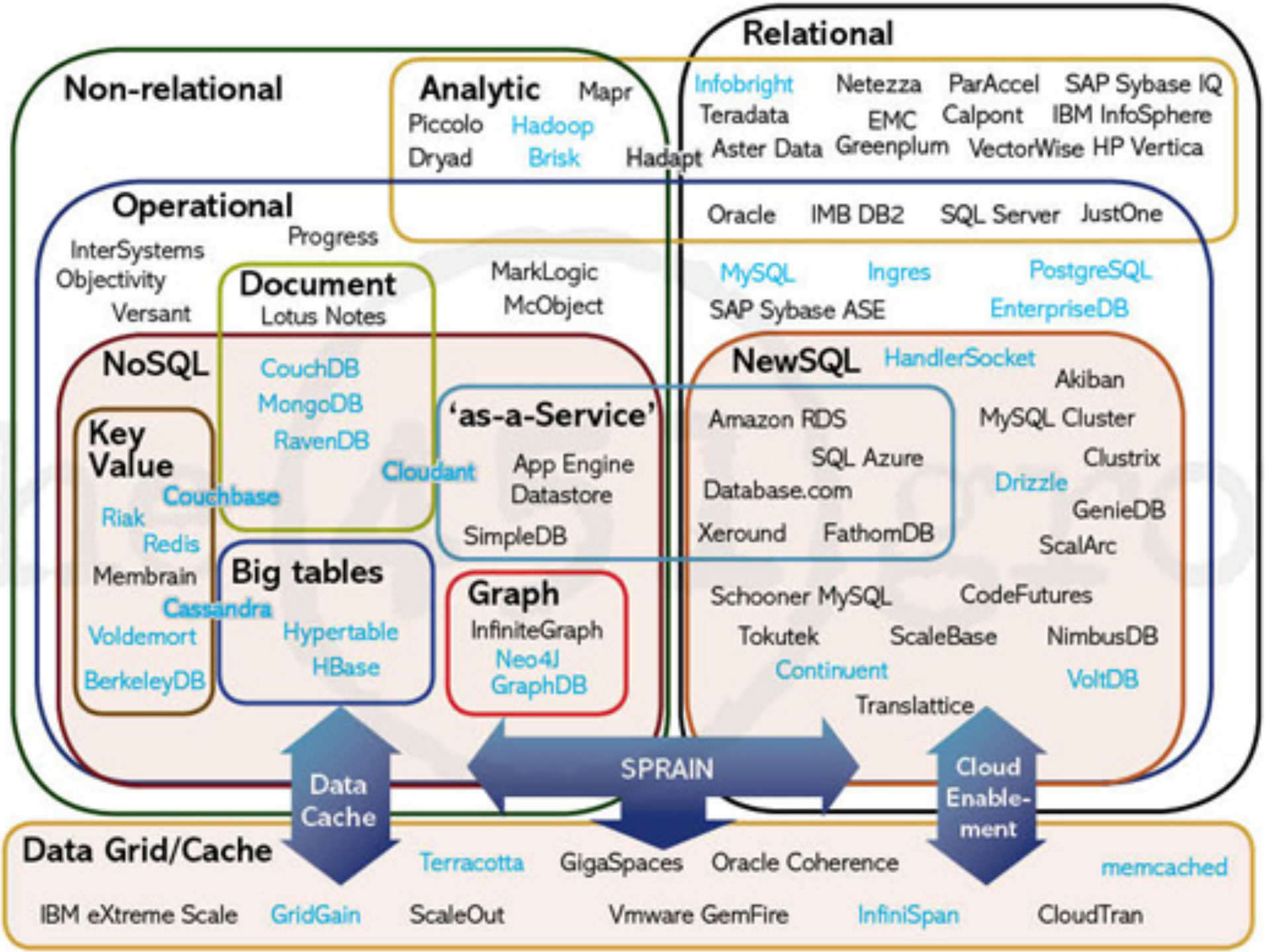
Dokumenten-orientierte Darstellung

Datenmanagement

- MongoDB stellt horizontale Skalierung für Datenbanken auf kostengünstigen, allgemein verfügbaren Rechensystemen bereit (mittels der sog. Sharding-Technik)
- Sharding verteilt Daten über mehrere physikalische Partitionen (shards) zur Vermeidung der Begrenzungen einzelner Rechner
- MongoDB sieht eine automatische Verteilung von Daten im Cluster vor, wenn Datenvolumina über Kapazität eines Einzelsystems hinaus gehen

NewSQL

- Neue Generation von relationalen DBMS
  - Stellen Performanz von NoSQL-Systemen bereit …
  - … für Online-Transaktionsverarbeitung (OLTP) mit Lese- und Schreiboperationen, und zwar unter Beibehaltung von ACID-Garantien relationaler DB-Systems
    - Kurze Transaktionen (keine Nutzerabbrüche)
    - Berühren kleine Mengen von Daten durch Indexzugriffe (keine ganzen Tabellen gelesen oder große verteilte Joins)
    - Anfragen sind wiederkehrend (auf neuen Daten)
  - Vermeidung von schwergewichtigen Reparaturmechanismen (wie z.B. Write-Ahead-Logs)
- Unterstützung von verteilten Clustern von Rechnern ohne gemeinsamen Speicher


Relational
Non-relational
Analytic
Mapr
Infobright
Netezza
ParAccel
SAP Sybase IQ
Teradata
EMC
Calpont
IBM InfoSphere
Piccolo
Hadoop
Aster Data
Greenplum
VectorWise
HP Vertica
Dryad
Brisk
Hadapt
Operational
Oracle
IMB DB2
SQL Server
JustOne
Progress
InterSystems
Objectivity
Versant
Document
Lotus Notes
MarkLogic
McObject
MySQL
Ingres
PostgreSQL
SAP Sybase ASE
EnterpriseDB
NoSQL
CouchDB
MongoDB
RavenDB
Cloudant
Key Value
Couchbase
Riak
Redis
Membrain
Cassandra
Voldemort
BerkeleyDB
'as-a-Service'
App Engine Datastore
SimpleDB
Big tables
Hypertable
HBase
Graph
InfiniteGraph
Neo4J
GraphDB
NewSQL
HandlerSocket
Akiban
Amazon RDS
MySQL Cluster
SQL Azure
Clustrix
Drizzle
Database.com
GenieDB
Xeround
FathomDB
ScalArc
Schooner MySQL
CodeFutures
Tokutek
ScaleBase
NimbusDB
Continuent
VoltDB
Translattice
Data Cache
SPRAIN
Cloud Enable-ment
Data Grid/Cache
Terracotta
GigaSpaces
Oracle Coherence
memcached
IBM eXtreme Scale
GridGain
ScaleOut
Vmware GemFire
InfiniSpan
CloudTran