

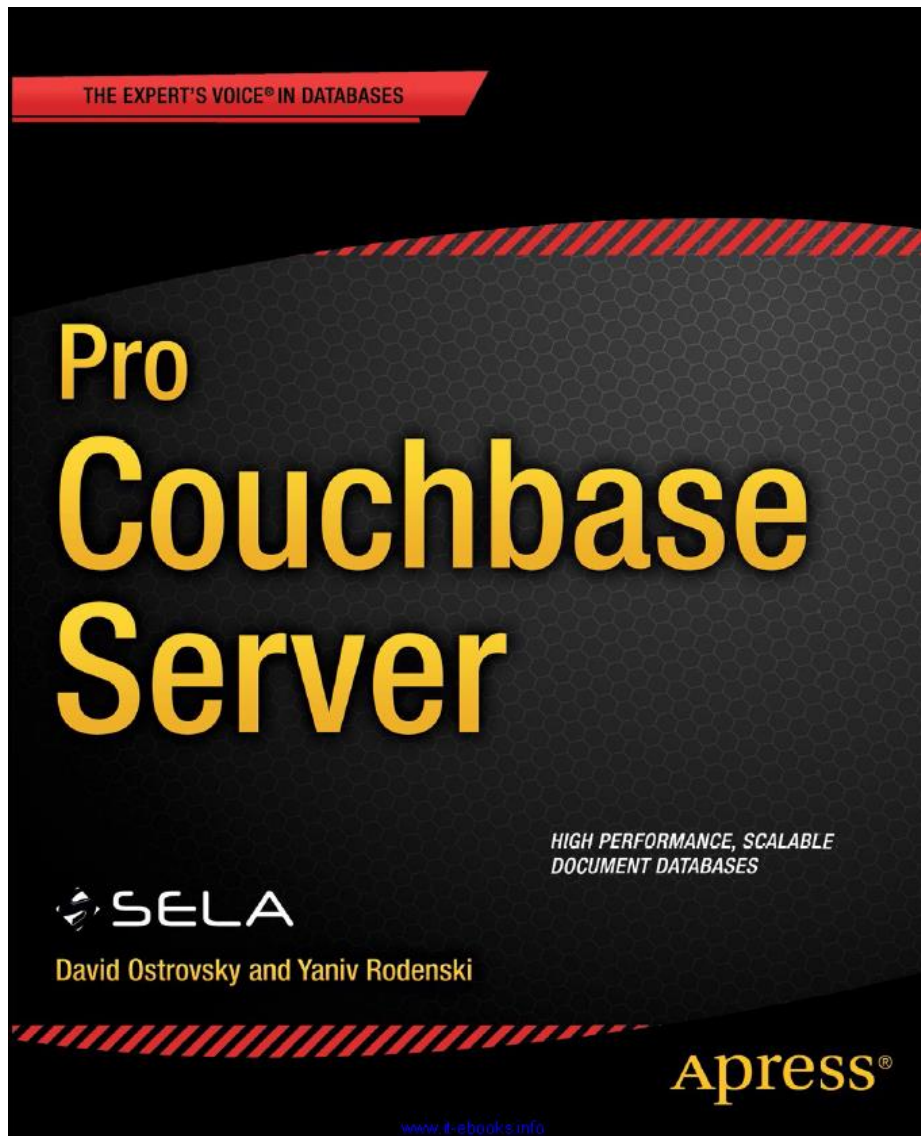


integrata  
cegos

# Couchbase Developer

Eine Dokumenten-orientierte Datenbank

- Die in diesem Seminar verwendete Werkzeuge und Frameworks sind Open Source
  - LPGL Lizenzmodell
- Dokumentation und Ressourcen stehen auch im Internet zur Verfügung
  - <https://apacheignite.readme.io/docs/what-is-ignite>



## Couchbase architectural advantages

NoSQL: Eine Kurzeinführung	6
Apache Ignite: Eine Übersicht	41
Einsatzbereiche von Apache Ignite	57
Das memory Grid	69
Anwendungen	89
Produktionsumgebung	102

# 1

## **NOSQL: EINE KURZEINFÜHRUNG**

1.1

## **BIG & FAST DATA**

- Definitionsversuche:
  - Physikalisch
    - Terabyte
  - System
    - Übersteigt den Bedarf eines normalen Speichermediums
  - Technisch
    - Datenverarbeitung beginnt, auf Grund der Größe Probleme zu bereiten
- Allerdings:
  - Keine wirklich eindeutige Definition erkennbar
  - Grenzen können sich verschieben

- Völlig verschiedene Bereiche:
  - DNS-Server für Internet
  - Indizierung von Web Seiten für Suchmaschinen
    - Google
  - Monitoring von Seitenzugriffen, aber auch Metrik-Informationen in großen Server-Systemen
  - Content Management und Waren-Systeme
    - Amazon
  - Social Media
    - Facebook



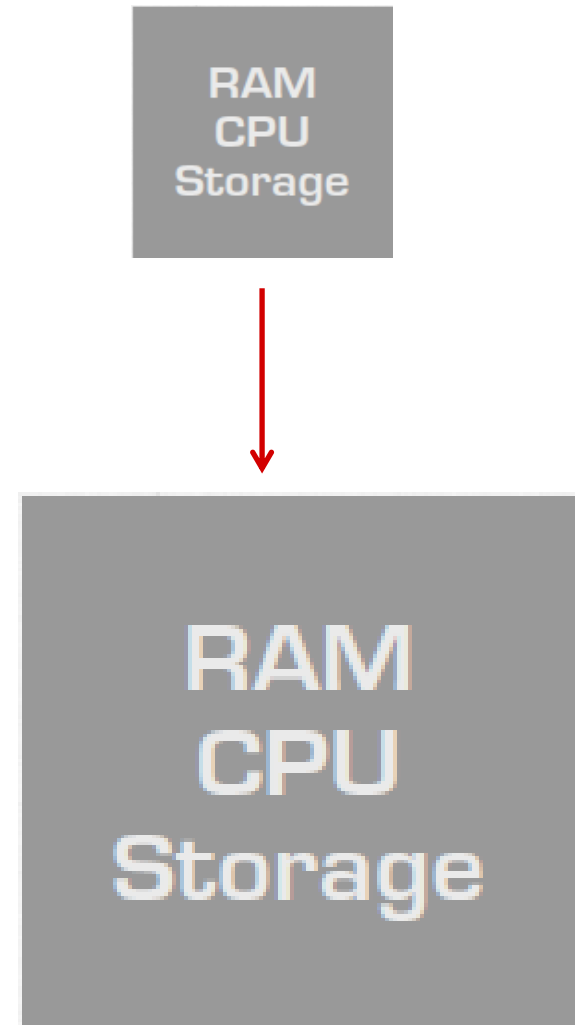
1.2

## **CLUSTERING**

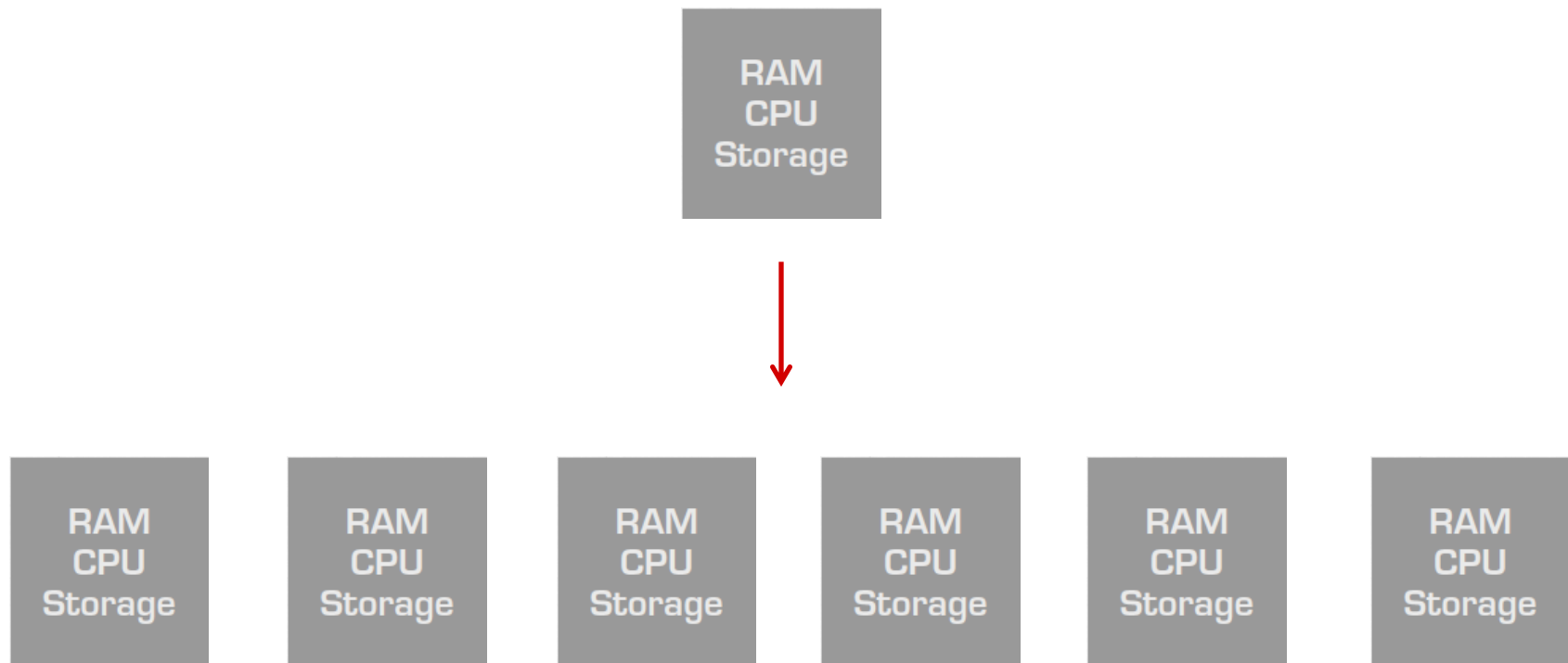
- Bei einem Cluster werden die Anwendungen auf identischen Servern betrieben
  - Betriebssystem
  - Laufzeitumgebungen
  - jeweils identische Versionen
- Ein Grid realisiert Anwendungen auf potenziell inhomogenen Rechnern
- Die Unterscheidung zwischen Grid und Cluster ist häufig fließend und deshalb nicht wirklich relevant
  - Im Folgenden wird einheitlich von Clustern gesprochen

- Zur Ablage und Auswertung von Daten sind selbst mächtige Server-Maschinen auf Dauer überfordert
- Die Datenhaltung wird deshalb auf mehrere Server verteilt
- Die Lastverteilung übernimmt ein simpler Load Balancer oder ein komplexer "Master"
  - Aus Client-Sicht heraus tritt der Cluster damit immer noch als eine Einheit auf
- Im Optimum ist der Cluster damit eine quasi unendliche Datensenke mit beliebiger skalierbarer Rechnerkapazität
  - Damit eine Vorstufe zur "Cloud"

- Grenzen sind klar definiert
  - Kosten
  - Aktuelle Hardware-Grenzen
  - Anforderungen an Ausfallsicherheit



- Grenzen
  - Ausfallsicherheit fordert Replikation über Netzwerk
  - System-Administration und Überwachung

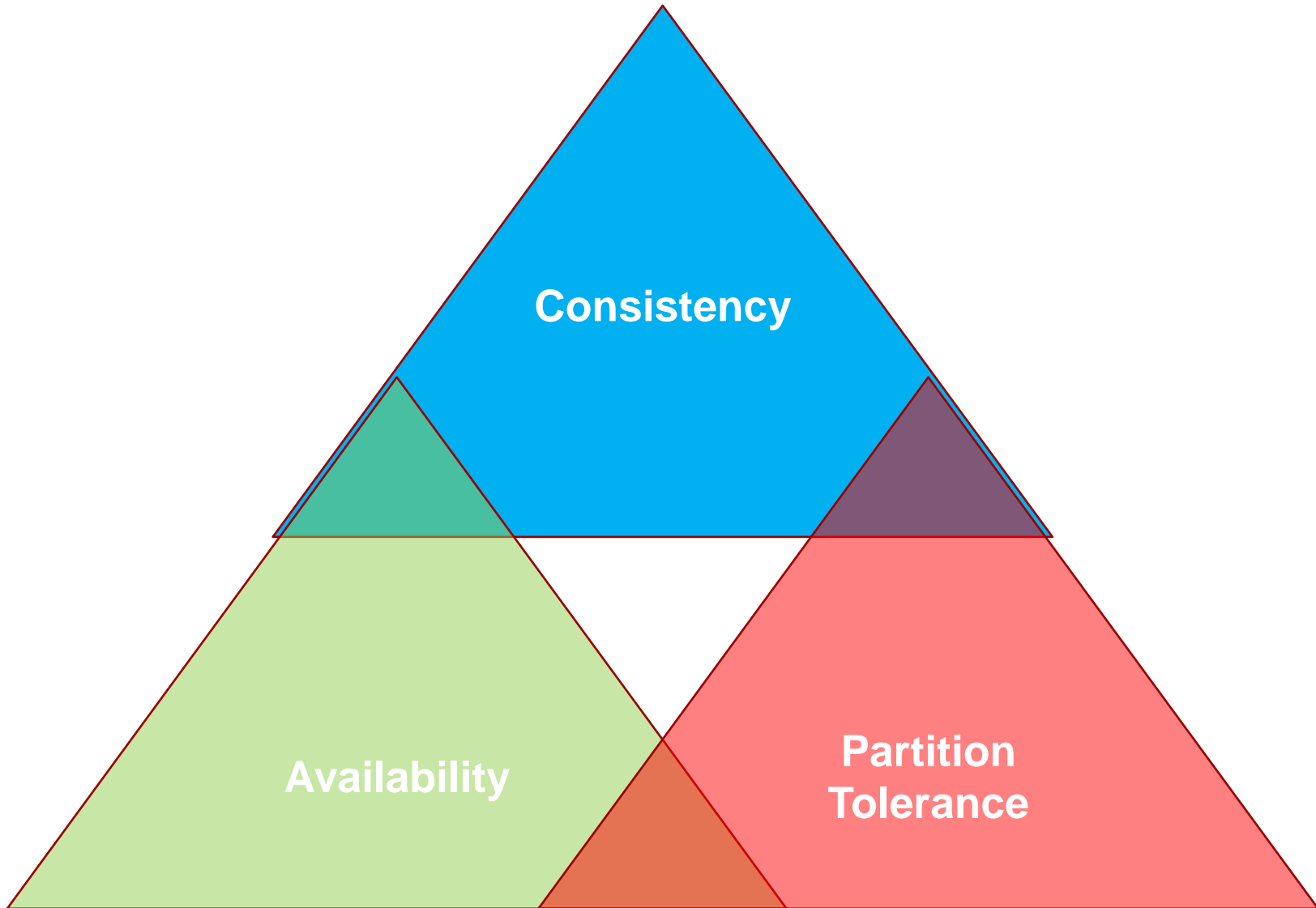


1.3

## **DAS CAP-THEOREM**

- Consistency
  - Alle Knoten haben jederzeit den selben Informationsstand
- Availability
  - Jeder Client bekommt garantiert Antwort
    - Solange zumindest ein Knoten im Cluster läuft
- Partition Tolerance
  - Das System toleriert
    - Den Ausfall von Knoten
    - Den Ausfall des Netzwerks zwischen Knoten

# Keine gemeinsame Schnittmenge!



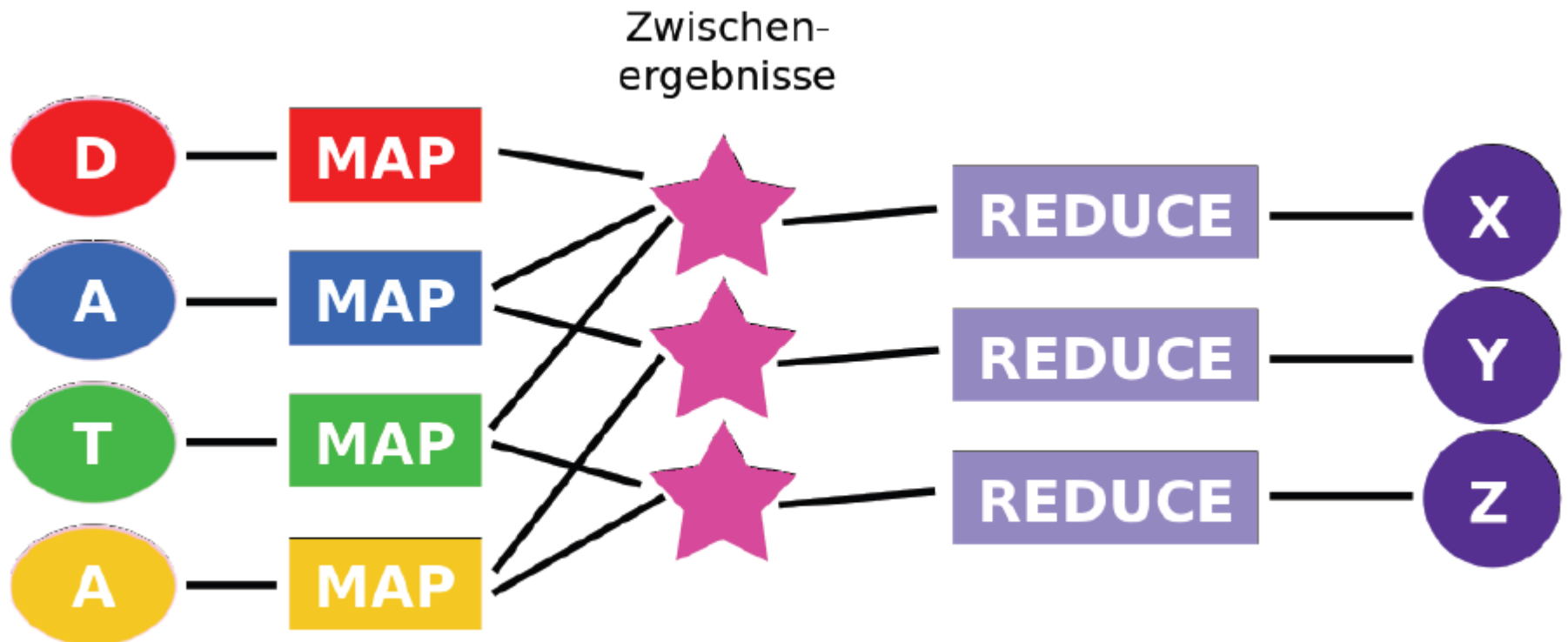


1.4

## **ANALYSE VON BIG DATA**

- Verlangt „neue“ Algorithmen
  - Aufteilung in Jobs
  - Diese werden parallelisiert
  - Operieren auf einem verteilten Datenbestand
- Alle genannten Unternehmen haben in den letzten 10 Jahren Algorithmen entwickelt bzw. eingesetzt
  - Und „netterweise“ der Open Source Community zur Verfügung gestellt
- Fast Data
  - Schneller Zugriff auf Daten-Selektionen
  - Schnelle Umsetzung neuer Abfrage-Anforderungen

- Bahnbrechende Entwicklung
  - Idee wird Google zugeschrieben
- Grundsätzliche Arbeitsweise
  - Operiert auf Mengen (Maps)
  - In einem ersten Schritt werden die Daten der Map in einer neuen Map aufbereitet
  - Diese wird in einem oder mehreren Reduce-Schritten zur Ergebnis-Struktur zusammengefasst



Quelle: <http://de.wikipedia.org>

1.5

# TRANSAKTIONALITÄT

- Atomic, Consistent, Isolated, Durable
- Features
  - Starke Konsistenz der Daten
  - Transaktionssteuerung
    - Transaction-Isolation
    - Bei Bedarf Zwei-Phasen-Commit
  - Relativ komplexe Entwicklung
- Relationale Datenbanken fokussieren auf ACID-Transaktionen

- Basically Available, Soft State, Eventual Consistency
  - Consistency
    - Alle Knoten haben **jederzeit** den selben Informationsstand
  - Eventually Consistent
    - Änderungen des Datenbestandes werden zeitlich versetzt an die anderen Knoten propagiert
    - "Eventual" nicht mit "eventuell" übersetzen!
- Features
  - Letzliche Konsistenz
  - Hohe Verfügbarkeit
  - Schnelligkeit
    - Bei Bedarf "Fire-and-forget"
  - Leichtere Entwicklung

1.6

## WAS IST NOSQL?



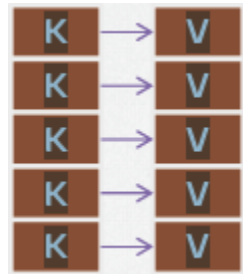
- No SQL!
  - Ursprüngliche Bedeutung
    - Etwas dogmatische Ablehnung relationaler Datenbank-Prinzipien
- Not only SQL
  - Schon deutlich abgeschwächt
  - Reduktion auf geeignete Problemstellungen
- No/ Not only relational
  - Der eigentlich richtige Begriff
  - Fokussierung auf die Daten-Modellierung, nicht auf die Abfragesprache

1.7

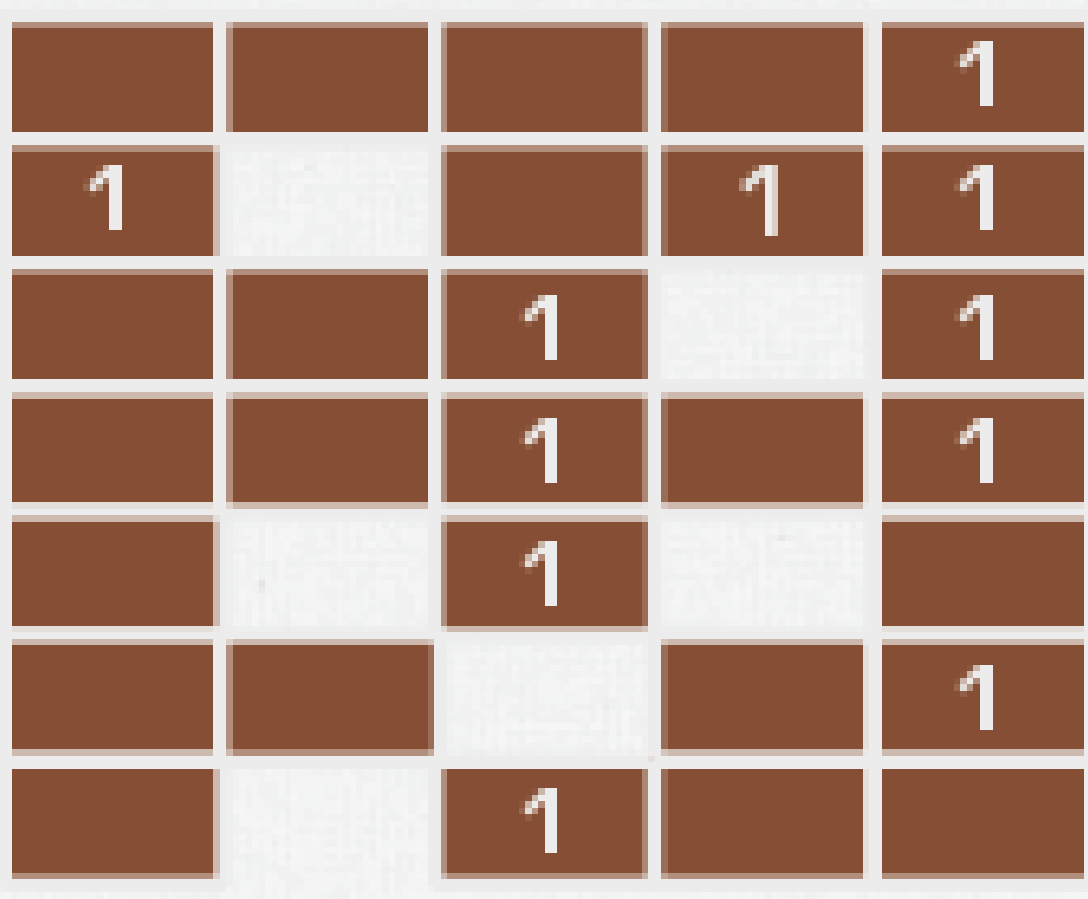
## **KLASSIFIZIERUNG**

- Key/Value
- Column-orientiert
- Dokumenten-orientiert
- Graphen-orientiert

- Ablage von Inhalten (=Value) unter einem eindeutigen Schlüssel (key)
  - Flache Struktur
  - Values werden als Byte-Array betrachtet
- Teilweise Verzicht auf Persistierung der Daten
  - Cache-Systeme
- Auslegung auf eine Vielzahl konkurrierender Zugriffe



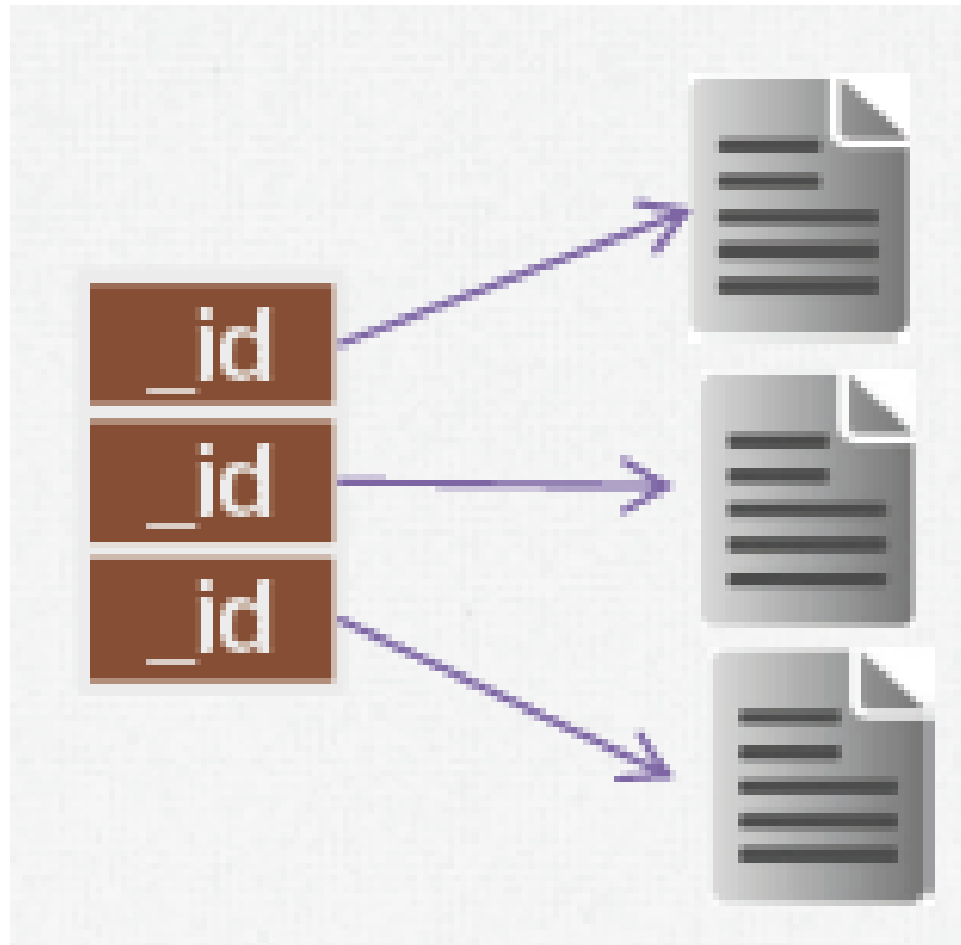
- Daten werden in Columns abgelegt
  - Eine Zeile besteht aber nicht zwangsläufig aus allen Columns
- Diese Columns können noch in Gruppen unterteilt werden
  - Vorteilhaft bei einer verteilten Datenhaltung
- Ausrichtung auf wirklich große Datenmengen



				1
1			1	1
		1		1
		1		1
		1		
				1
		1		

- Ablage der Daten in strukturierten Dokumenten
  - Nicht unbedingt XML!
  - Wesentlich weiter verbreitet ist JSON
    - Bzw. das binäre BSON





- Ablage der Daten in
  - Knoten
  - Beziehungen
  - Attribute
- Knoten und Beziehungen haben Attribute
- Knoten sind mit Beziehungen verknüpft und umgekehrt
- „Whiteboard-Friendly“
  - „Was Sie auf einem Whiteboard zeichnen können, können Sie in einer Graph-DB ablegen.“



1.8

# DATENMODELLIERUNG

- Daten werden abstrahiert modelliert
  - Klassenorientierte Programmierung
    - Java, C++/C#, ...
  - Tabellen-Schemata
  - XML-Schema
- Eine Umwandlung von Datentypen erfordert ein Umkopieren
  - „Migration“
- Damit sind Entwicklungszyklen und die Zeiten für die Einführung geänderter Features relativ lang

- „Duck Typing“: Nur vorhandene Eigenschaften und implementiertes Verhalten zählen
  - „Was gelb ist, quakt und watschelt ist eine Ente“
- Dies ist vorwiegend die Domäne von dynamischen Skript-Sprachen
  - Ruby, Python, JavaScript
- Ein Umkopieren von Daten ist nur noch in Ausnahmefällen notwendig
- Damit wird eine agile Software-Entwicklung unterstützt

- Was ist...
  - Besser?
  - Richtig?
  - Komfortabel?
- Leider keine pauschale Antwort möglich
  - „Wer lebend den Raum verlässt hat gewonnen“
- Aber:
  - Der Trend der letzten Jahre geht immer mehr Richtung Dynamik

# DIE COUCHBASE



# FEATURES

- Eine dokumenten-orientierter Key-Value-Store
- Horizontale Skalierung im Ring-Cluster
  - Master-less
  - Admin-less
  - Konfigurierbare Ausfallsicherheit durch Replikation
- Administration und Überwachung durch Web-Konsole
- Distribution

- Ablage der Daten in "Buckets"
  - Entsprechen einer Tabelle
  - Im Objekt-orientierten Sinne einer Tabelle
  - Ein simpler Bucket entspricht einem Key-Value-Store
- Values sind JSON-Strukturen
  - Schema on Read
  - Daten-Modellierung ist damit Dokumenten-orientiert

- Couchbase wird meistens als CP-System betrieben
  - Bei großen Systemen mit mehreren Rechenzentren wird auf AP und damit eine BASE-Architektur umgestellt

- Couchbase-Views
  - Parametrisierte Map-Reduce-Algorithmen
- N1QL
  - Non-First Normal Form Query Language
  - Angelehnt an SQL
    - Keine Unterstützung des vollen SQL-Standards möglich
    - Dafür Ergänzungen, die aus der Dokumenten-Modellierung heraus resultieren

## **WEITERFÜHRENDE RESSOURCEN**



# Couchbase

Consistent. Elastic. Scalable. Always available. Data storage.

Kelum Senanayake

## N1QL: A PRACTICAL GUIDE

**N1QL = SQL + JSON**



Gerald Sangudi  
Marco Greco  
Isha Kandaswamy  
Eben Haber  
Keshav Murthy

Sitaram Vemulapalli  
Johan Larson  
Prasad Varakur  
Manuel Hurtado  
Tai Tran

*Foreword by Gerald Sangudi, father of N1QL*