Ausblick

Alternativen zu relationalen Datenbanken

Aktuelle Trends: Anwendungen

- Nicht-Standard-Anwendungen
  - Geo-Informationssysteme
  - CAD-Systeme
  - Naturwissenschaftliche Anwendungen (Scientific Databases)
  - Multimedia / Bildverarbeitung
  - Digitale Bibliotheken
- Web 2.0
  - Soziale Netzwerke, z.B. Facebook, Twitter
  - Benutzer produzieren auch Daten (Blogs, Wikipedia)
- Electronic Commerce
  - Elektronische Marktplätze, z.B. Amazon, Ebay
  - Wertschöpfungsketten unternehmensübergreifend (Virtual Enterprise)
- Semantic Web / Information Extraction
  - Internet als einzige große Datenbank

Aktuelle Trends: Technologische Entwicklungen

- eingebettete Datenbanken (Mini Databases)
- Konvergenz mit objektorientierten Systemen (Sprache, Datenmodelle)
- Big Data / Datenwachstum (Petabyte-Zeitalter)
- Data Management on New Hardware
  - In Memory Databases
  - Flash Memory als Alternative zur Disk
- Replizierte Daten
  - Speicherung auf mehreren Knoten (d.h. Computern) im Netz
  - anwendbar für alle Arten von Datenbanksystemen
- Horizontale Skalierbarkeit
  - Systeme auf wachsende Datenmengen ausgelegt
  - Dynamische Bedarfsanpassung durch Hinzunahme zusätzlicher Knoten

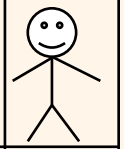
Wie können relationale Datenbanken ersetzt / ergänzt werden ?

NoSQL

Objektorientierte Datenbanken

Warum reichen SQL-Datenbanken nicht aus?

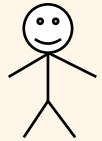
Tabelle aus relationaler DB mit scheinbar gleichartiger Struktur der Datensätze (Beispiel Adressverwaltung)

Schmitz	Hannes	Köln	Steinstr. 11	0221489090			
Meier	Hans	München					
Goldmann	S.			0172123456			
Schleicher	Helga				16.09.1948		
Kramer	Nils	Essen	Aststr. 12		21.11.1988		
Rust	Sonja			034196385		0163654321	

- starre Satzstruktur zu wenig flexibel für unterschiedliche und beliebige Einträge zur Verwaltung von Kontakten
- viele, oft nicht benötigte, Attribute → große Spaltenanzahl → viele Nullwerte
- *Beispiel*: Geburtsdaten zumeist nur für Freunde abspeichern

Warum reichen SQL-Datenbanken nicht aus?

Datensätze mit unterschiedlicher Satzstruktur

Schmitz	Hannes	Köln	Steinstr. 11	0221489090
Meier	Hans	München		
Goldmann	S.		0172123456	
Schleicher	Helga	16.09.1948		
Kramer	Nils	Essen	Aststr. 12	21.11.1988
Rust	Sonja	034196385	0163654321	

NoSQL

- NoSQL = **N**ot **o**nly SQL
- Datenmodelle bilden Erweiterung zu Datenbanken mit relationalem Datenmodell (not only)
  - Kombination mit SQL-Datenbanken, z.B. NoSQL zur Datenarchivierung
- nicht-relationale Datenmodelle
  - Dokumentenorientiert
  - Key Value Stores
  - Graphenorientiert
  - Spaltenorientierte Datenbanken
  - andere Datenmodelle: objektorientiert, XML
- Schemafreiheit
  - zusätzliche Informationen in Datensätze enfügen
  - vermeidet das Problem der Schema-Evolution
- verteilte Speicherung
- Nähe zum Web (Schnittstellen für Skriptsprachen)

Konsistenz in relationalen und NoSQL-DBS

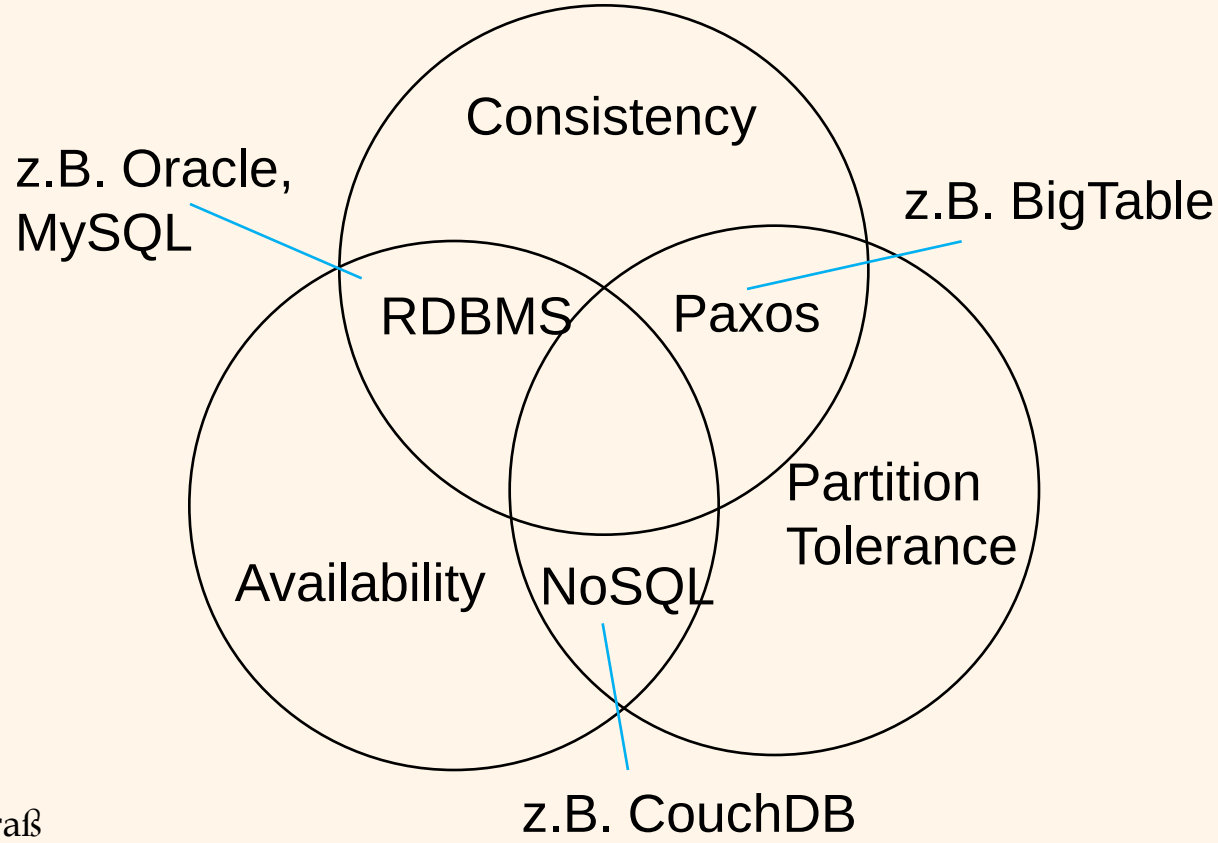
- Logischer Datenbankentwurf in relationalen DBS
  - Normalisierung
  - Integritätssicherung: referentielle Integrität, semantische Integrität
  - Transaktionskonzept: ACID-Paradigma
- Konsistenz bei NoSQL
  - Abschwächung in verteilten Datenbanken
  - Herstellung der Konsistenz nur zu verschiedenen Zeitpunkten, also „letztendlich“ (eventual consistency)
- Anwendungen im Web akzeptieren schwächere Konsistenz
  - E-Commerce
  - Beispiel: Aktualisierung eines Produktkatalogs in verteilter Datenbank – nicht als ACID-Transaktion notwendig

CAP-Theorem

- Autor: Eric Brewer, University of California (2000)

Verteilte Datenbanken erfüllen höchstens zwei der drei CAP-Eigenschaften

- CAP = Consistency, Availability, Partition Tolerance



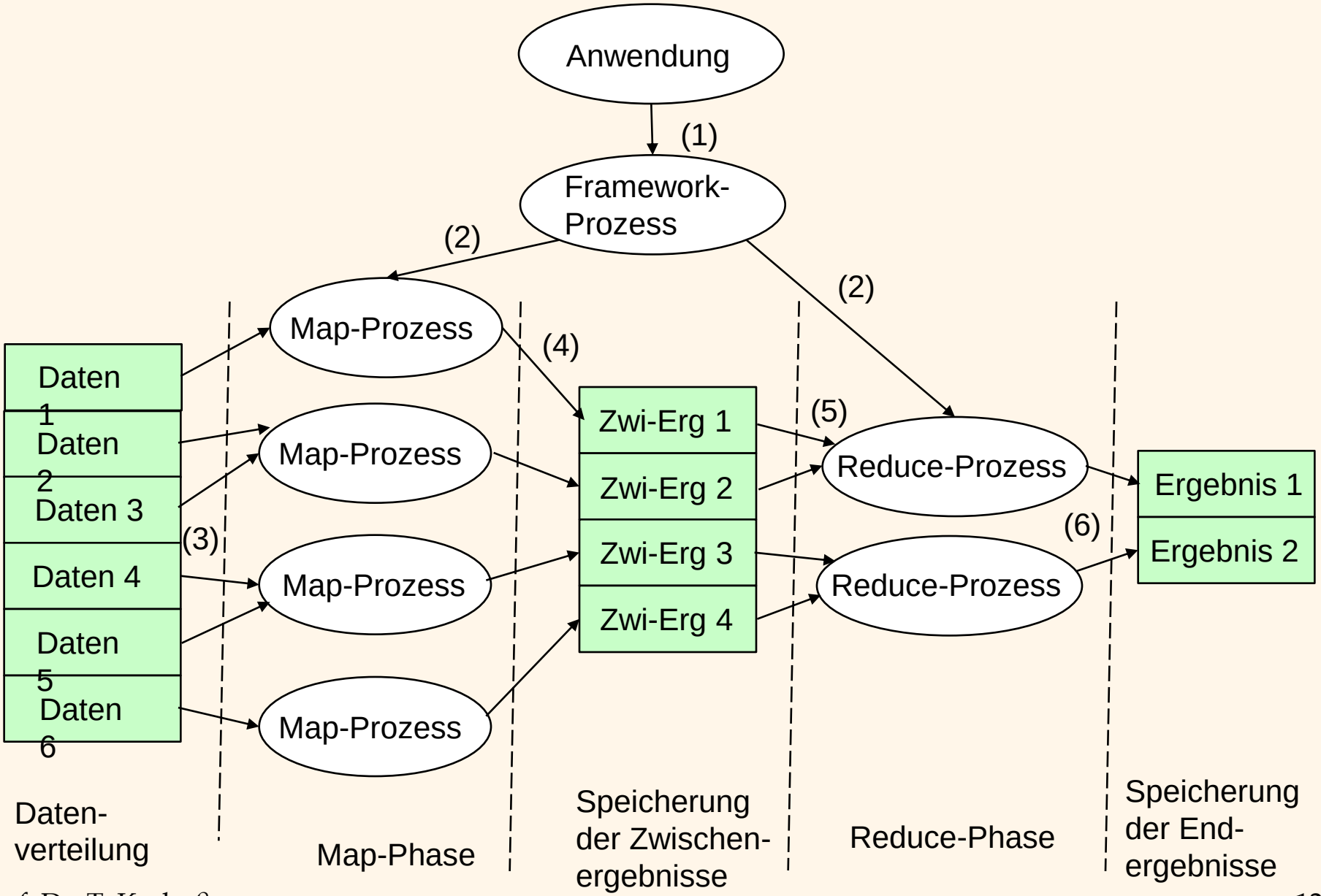
BASE-Konzept

- Konsequenz des CAP-Theorems
  - Abschwächung der ACID-Eigenschaften
  - Priorität auf Verfügbarkeit und Ausfalltoleranz
- BASE = Basically Available, Soft State, Eventual Consistency
- Eventual Consistency
  - Datenkonsistenz letztendlich garantiert (eventual)
  - Zeitpunkt der Einhaltung der Konsistenz nicht a-priori garantiert (vgl. ACID: Konsistenz am Ende der Transaktion)
  - Konistenzkontrolle durch MVCC-Protokoll (Multi Version Concurrency Protocol)
  - Vorteil: beschleunigtes Laufzeitverhalten
- Basically Available
  - ohne zeitliches Blockieren von Teilen einer Datenbank
  - NoSQL-DBS grundsätzlich immer verfügbar
- Soft State
  - Daten nicht persistent zwischen den konsistenten Zuständen der Datenbank (vs. Durability)

MapReduce

- Ermittlung von Informationen aus großen verteilt gespeicherten Datenmengen durch Parallelverarbeitung
- Google (2004): neues Abfragemodell aus bestehenden Konzepten funktionaler Programmiersprachen (z.B. LISP, Haskell)
- 2 Phasen:
  - ***Map*-Phase**: Untersuchung des Datenbestandes anhand fachlicher Kriterien und temporäre Speicherung des Ergebnisses
  - ***Reduce*-Phase**: Zusammenfassung der Zwischenergebnisse anhand fachlicher Kriterien zum gesuchten Ergebnis
- Bereitstellung der fachlichen Logik durch Anwender
  - ***Map*-Funktion**: ermittelt zu den vorgegebenen Key-/Value-Paaren neue, temporäre Key-/Value-Paare
    formal: `Map(in_key, in_value) -> list(out_key, intermediate value)`
  - ***Reduce*-Funktion**: transformiert Zwischenergebnisse anhand von Gruppierungen/Aggregation in eine Ergebnisliste
    formal: `Reduce(out_key, list(intermediate_value)) -> list(out_value)`
- Ausführung der Aufgaben durch Framework (z.B. Hadoop)

MapReduce Algorithmus



Beispiel MapReduce

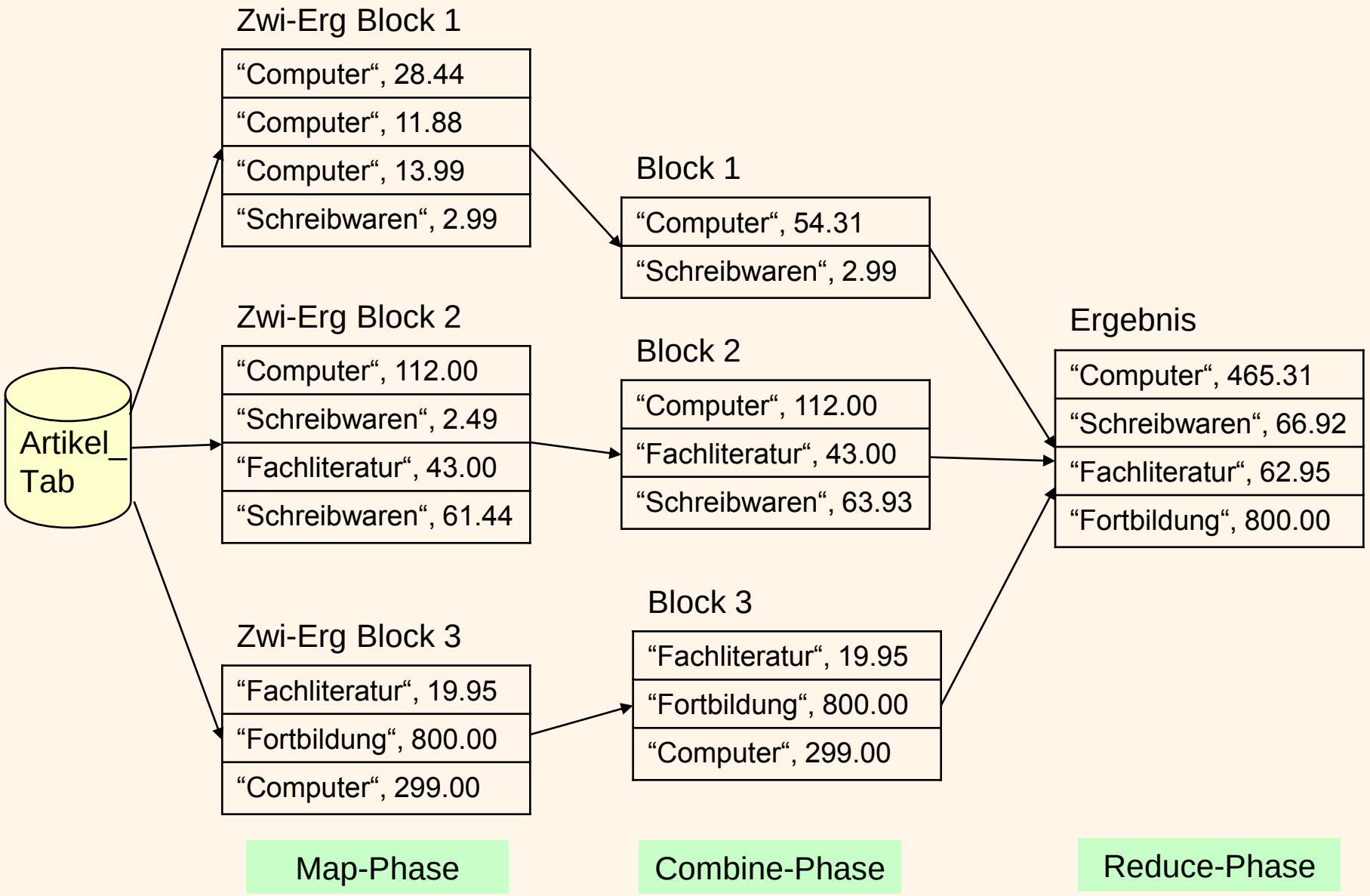
Ermittle die Kosten pro Kostenart

Datum	Kostenbezeichnung	Kostenart	Preis
29.10.2005	Druckerpatrone schwarz	Computer	28.44
01.12.2005	DVD Rohling 10 St.	Computer	11.88
17.11.2004	CD Rohlinge 50 St.	Computer	13.99
01.12.2004	Druckerpapier	Schreibwaren	2.99
29.10.2005	Festplatte	Computer	112.00
24.10.2005	Druckerpapier	Schreibwaren	2.49
02.12.2005	Java-Einführung	Fachliteratur	43.00
24.10.2005	Overhead Folien	Schreibwaren	61.44
07.01.2006	Java-Die Progr.-sprache	Fachliteratur	19.95
14.04.2006	Java-Schnellkurs	Fortbildung	800,00
04.03.2003	Drucker	Computer	299.00

- Anfrage in SQL:

```
SELECT   Kostenart, SUM(Preis)
FROM     Artikel_Tab
GROUP BY Kostenart;
```

Beispiel MapReduce (Forts.)



CouchDB als Beispiel einer NoSQL-DB

- Historie
  - Autor Damien Katz 2005
  - Weiterentwicklung bei IBM 2008-09
  - seit 2008 Apache-Projekt unter Apach 2.0 Lizenz
- Datenmodell
  - Speicherung der Daten in Dokumenten (vgl. Datensatz in RDB)
  - Reine Textinformationen oder Bilder, Video-/Audio-Dateien
- JSON
  - Inhalt eines Dokuments entspricht JSON-Objekt
  - JSON einfache Beschreibungssprache aus JavaScript-Welt
  - Beispiel:

Key/Value

Liste

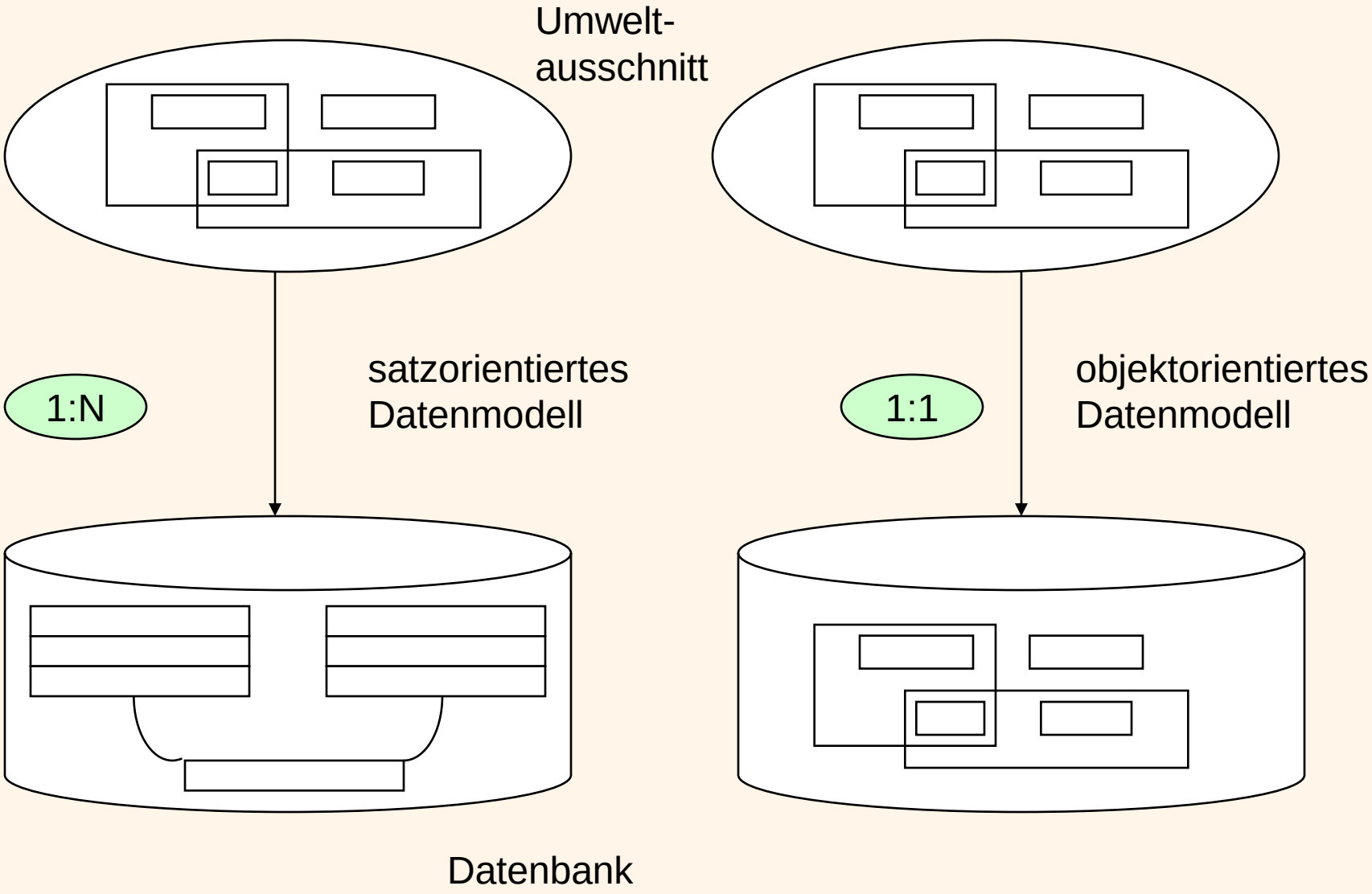
Array

```
{
  "Nachname": "Meier",
  "Vornamen": ["Tobias", "Eugen"],
  "Alter": 32,
  "Telefon": {"Handy":  "017798765" ,
              "privat": "034154321" }
}
```

Eigenschaften von CouchDB

- CouchDB API
  - Zugriff über das RESTful JSON API: GET, PUT, DELETE, POST Requests des HTTP-Protokolls zur Erteilung von Verarbeitungsanweisungen
  - Zugriff über Kommandozeilen-Tool cURL oder Futon-Browser als GUI
- JavaScript
  - Programmierung von Views unter Verwendung der MapReduce-Technologie
  - Map- und Reduce-Funktionen in JavaScript als Design-Dokumente im JSON-Format gespeichert
  - Bibliotheken und Clients für viele andere Sprachen (z.B. PHP, Python) analog zu RDBMS
- Erlang
  - funktionale Programmiersprache zur Verarbeitung nebenläufiger und verteilter Prozesse
  - genutzt zur Entwicklung von CouchDB
- Datenreplikation und Multiversion Concurrency Control

Objektorientierung



Neue Anforderungen

- Modellierung hochstrukturierter Informationen
  - beliebig komplexe Objekte
  - Versionen: Alternativen, Revisionen, Varianten
  - Benutzerdefinierte Datentypen und Typhierarchien
- Modellierung semantischer Beziehungen, z.B. räumliche Beziehungen
- Modellierung von typspezifischem Verhalten
  - Operatoren und Methoden zum Umgang mit komplexen Objekten (z.B. topologische Operatoren für Geo-Objekte)
- Überwindung des Mismatch zwischen DB und Programmier-sprache (Java)
- Lange Transaktionen (z.B. bei CAD)
- Komplexe Integritätsbedingungen (zeitbezogen, räumlich)

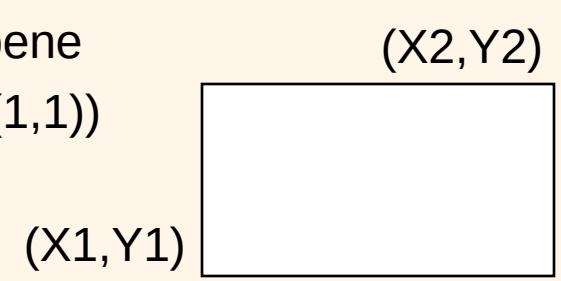
Objektorientierte Konzepte für eine Datenbank

- Objektidentität
  - Surrogate, systemdefiniert, unveränderbar
  - Identität vs. Gleichheit
  - Objektreferenzen, Navigation entlang der Referenzen
- komplexe Objekte
  - Definition komplexer Objekttypen mittels Typkonstruktoren
  - Operatoren für komplexe Objekte (z.B. Geometrie)
- Kapselung
  - Öffentliche Schnittstelle
  - Implementierung der Methoden verborgen
  - Abstrakte Datentypen (ADT), erweiterbar durch Benutzer
- Typen und Klassen
  - Typ als Beschreibung von Objekten mit gleicher Struktur / Verhalten
  - Klasse als Menge von Objekten (Container)
- Vererbung
  - Anordnung von Objekttypen und Klassen in Generalisierungs-/Spezialisierungshierarchie
  - Überladen (Overloading) und spätes Binden (Late Binding) von Methoden

Anwendungsbeispiel: Verwaltung räumlicher Objekte

Beispiel: Darstellung von Rechtecken in der Ebene

Finde alle Rechtecke, die das Rechteck ((0,0),(1,1)) schneiden!



Relationenmodell: Tabelle

R-ECK (RNR, X1, Y1, X2, Y2: INTEGER)

```
SELECT RNR FROM R-ECK
WHERE       (X1 > 0 AND X1 < 1 AND Y1 > 0 AND Y1 < 1)
            OR (X1 > 0 AND X1 < 1 AND Y2 > 0 AND Y2 < 1)
            OR (X2 > 0 AND X2 < 1 AND Y1 > 0 AND Y1 < 1)
            OR (X2 > 0 AND X2 < 1 AND Y2 > 0 AND Y2 < 1)
```

Objektorientiertes Modell: Abstrakter Datentyp (ADT)

ADT BOX mit Funktionen INTERSECT, CONTAINS, AREA usw.

R-ECK (RNR: INTEGER, Beschr: BOX)

```
SELECT RNR FROM R-ECK
WHERE INTERSECT (Beschr, (0,0,1,1))
```

Objektrelationale Konzepte im SQL-Standard

- Typkonstruktoren
  - geschachtelte Anwendung: beliebig komplexe Datentypen
- Benutzerdefinierte Datentypen
  - Distinct-Typen: Kopien eines Basisdatentyps unter eigenem Namen
  - Strukturierte Typen: Attribute und Methoden, Subtyp-Beziehung möglich
- Benutzerdefinierte Casts (Typumwandlungen)
- Benutzerdefinierte Ordnungen
- Typisierte Tabellen
  - basieren auf strukturiertem Typ
  - Tabellenhierarchien (Subtabelle steht in Untermengenbeziehung zur Supertabelle)
- Typisierte Sichten
  - Basieren auf strukturiertem Typ (liefern typisierte Tabelle)
  - Sichtenhierarchien

Beispiel: Objektrelationale Konzepte in Oracle

- Definition von Objekttypen (CREATE TYPE)
- Definition von Objekttabellen (CREATE TABLE .. OF ..)
- Objekttypen und Referenzen (REF)
- Methoden in Objekttypen
- Collections (VARRAY & Nested Table)
- Typvererbung (UNDER)
- Polymorphismus (Overriding, Overloading)
- Funktionen und Prädikate für Objekte (REF, DEREF, TREAT, IS OF)

Objektorientierte Datenbank-Technologien

- Weiterentwicklung relationaler zu objektrelationalen DBMS (vgl. auch SQL:1999)
  - benutzerdefinierte Datentypen und Datenbankroutinen
  - typisierte Tabellen
  - gegenwärtig noch große Heterogenität der DBMS
  - Pakete für benutzerdefinierte Erweiterung (z.B. multimediale Datentypen)
- Gateway-Ansatz zur Integration von Persistenz in objekt-orientierten Anwendungen
  - Middleware-Lösung zur Nutzung nicht-objektorientierter Datenquellen
  - Intelligentes Cache-Management und Entwicklungswerkzeuge (z.B. Mapping Objektmodell ⇔ relationales Datenmodell)
  - Persistenz-Frameworks: Hibernate, EclipseLink auf der Basis der Java Persistence API (JPA)
- Objektorientierte Datenbanksysteme
  - Produkte: Versant, db4o, ObjectDB

Fazit

- Man kann ALLES in einem relationalen DBMS (+ einer konventionellen Programmiersprache) ausdrücken:
  ABER: In vielen Fällen gibt es bessere Wege zum Ziele

BESSERE EFFIZIENZ
- Entwurfs-, Entwicklungs- und Wartungseffizienz
- Ausführungseffizienz

verbesserte Datenmodellierung (umfassender, präziser)

höhere Flexibilität durch schwache Strukturen und Schemafreiheit

verbesserte Verhaltens-modellierung (Business-Logik in der DB)

Implementation von Anfragen mittels Datenverteilung und Parallelisierung (MapReduce)

OODBS + NoSQL-DB

RDBS