Big-Data-Technologien

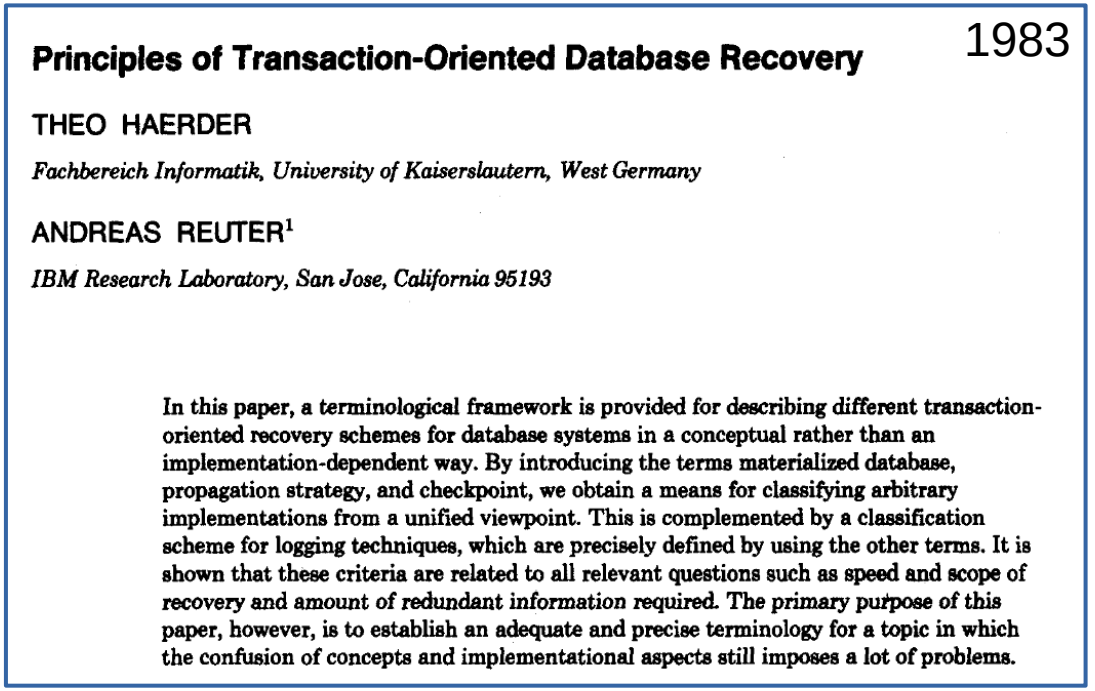
Kapitel 10: Eigenschaften verteilter Datenbanksysteme

Hochschule Trier
Prof. Dr. Christoph Schmitz

Wiederholung: ACID

- **Transaktionen** (nicht nur) in relationalen Datenbanken sind:
  - **A** atomar
  - **C** konsistenzerhaltend
  - **I** isoliert
  - **D** dauerhaft

... theoretisch :-(



1983

Principles of Transaction-Oriented Database Recovery

THEO HAERDER

Fachbereich Informatik, University of Kaiserslautern, West Germany

ANDREAS REUTER[1]

IBM Research Laboratory, San Jose, California 95193

In this paper, a terminological framework is provided for describing different transaction-oriented recovery schemes for database systems in a conceptual rather than an implementation-dependent way. By introducing the terms materialized database, propagation strategy, and checkpoint, we obtain a means for classifying arbitrary implementations from a unified viewpoint. This is complemented by a classification scheme for logging techniques, which are precisely defined by using the other terms. It is shown that these criteria are related to all relevant questions such as speed and scope of recovery and amount of redundant information required. The primary purpose of this paper, however, is to establish an adequate and precise terminology for a topic in which the confusion of concepts and implementational aspects still imposes a lot of problems.

ACID in der Praxis

- ANSI SQL definiert verschiedene **Isolationsebenen**:

ANSI isolation level	Dirty reads	Non-repeatable reads	Phantom reads
Serializable	No	No	No
Repeatable read	No	No	Yes
Read committed	No	Yes	Yes
Read uncommitted	Yes	Yes	Yes

- Gründe
  - Performance
  - Deadlocks

Anforderungen an verteilte Datenbanksysteme

- **Konsistenz** (Consistency)
  - jede Leseoperation liefert den aktuellsten Zustand
- **Verfügbarkeit** (Availability)
  - jede Anfrage wird beantwortet
- **Partitionstoleranz** (Partition Tolerance)
  - System kann mit verlorenen Nachrichten umgehen

Achtung, zwei verschiedene Konsistenzbegriffe!

- Relationale DB, ACID:

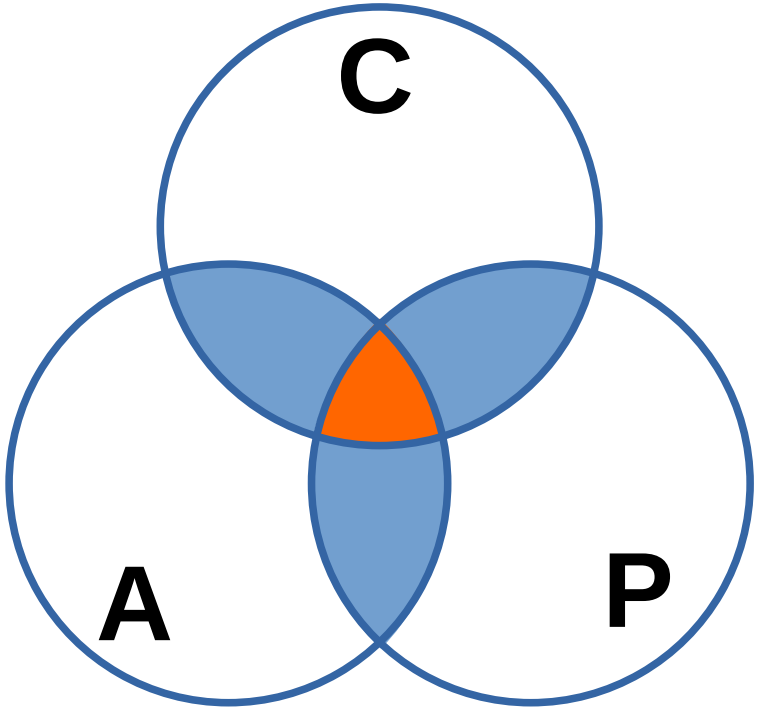
  fachlich sinnvoller Zustand

- Verteilte DB, NoSQL:

  Übereinstimmung über Zustand

CAP-Theorem (Brewer, 2000)

Ein verteiltes System kann höchstens zwei der drei Eigenschaften **Konsistenz**, **Verfügbarkeit** und **Partitionstoleranz** garantieren.



CAP-Theorem

- Netzwerkausfälle sind unvermeidbar
  → **Partitionstoleranz** muss sein

Verfügbarkeit

- Hinnehmen von „unsauberen“ Schreiboperationen
- Reparieren zu einem späteren Zeitpunkt

→ **Konsistenz** aufgeben

Konsistenz

- Protokolle zur Absicherung von Schreiboperationen
- Im Zweifelsfall Operation zurückweisen

→ **Verfügbarkeit** aufgeben

CAP-Theorem: Einschränkungen

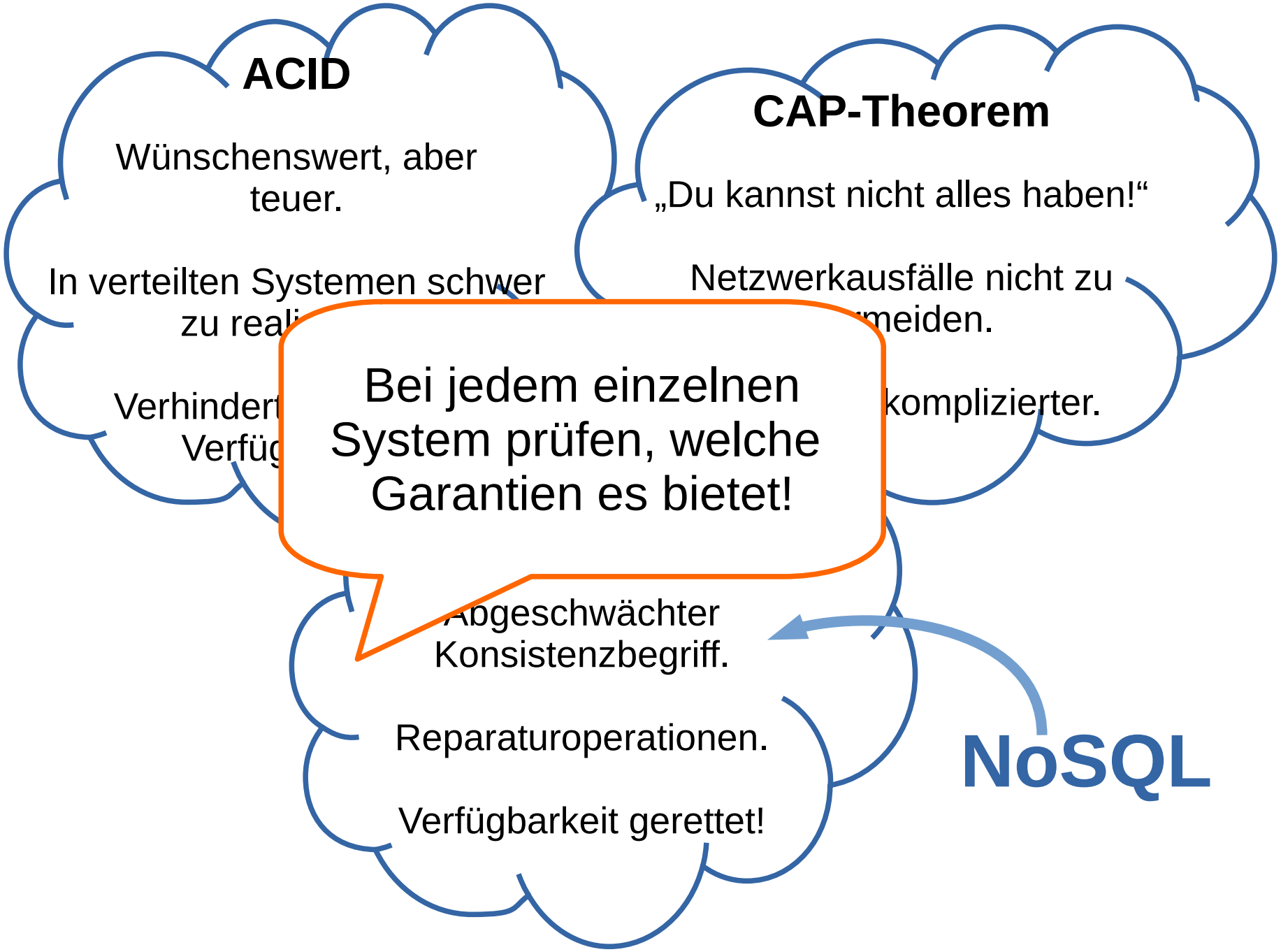
- Eng definierte Begriffe von **C**, **A** und **P**
- Eingeschränktes **Datenmodell** (ein Register)
- Nur **bestimmte Art** von Fehlern
- …

→ die Realität ist **viel komplizierter**!

Noch ein Buzzword: BASE

- **B**asically **A**vailable, **S**oft state, **E**ventually consistent

- **Basically Available:** System antwortet auf jeden Fall „irgendetwas“
- **Soft State:** Zustand kann sich spontan ändern, z. B. bei Reparaturoperationen
- **Eventually Consistent:** im Ruhezustand wird das System irgendwann konsistent


ACID
Wünschenswert, aber teuer.
In verteilten Systemen schwer zu reali
Verhindert
Verfüg
CAP-Theorem
„Du kannst nicht alles haben!“
Netzwerkausfälle nicht zu meiden.
komplizierter.
Bei jedem einzelnen System prüfen, welche Garantien es bietet!
Abgeschwächter Konsistenzbegriff.
Reparaturoperationen.
Verfügbarkeit gerettet!
NoSQL

Ressourcen

Dynamo: Amazon's Highly Available Key-value Store

Giuseppe DeCandia, Deniz Hastorun, Madan Jampani, Gunavardhan Kakulapati, Avinash Lakshman, Alex Pilchin, Swaminathan Sivasubramanian, Peter Vosshall and Werner Vogels

Amazon.com