

```
SQL> SELECT compressed(know_how)
      2 FROM database_community
      3 FOR user_level='BEGINNER' ;
```

12 presentations selected.

Einsteiger-Stream auf der DOAG-Datenbank-Konferenz 2019

Montag, 3. Juni

10:15	Codd & ACID – Ein Ausflug in die Datenbank-Theorie und -Geschichte	Markus Flechtner, Trivadis GmbH
11:15	Transaktionskonzept und Umsetzung	Martin Klier, Performing Databases GmbH
12:15	Überblick und Architektur	Martin Klier, Performing Databases GmbH
14:15	DB-Selbstverwaltung – das Data Dictionary	Markus Flechtner, Trivadis GmbH
15:15	Die Herausforderungen einer Datenbank	Martin Schmitter, RWE Supply+Trading GmbH
16:15	Oracle Multitenant	Ulrike Schwinn, Oracle Deutschland B.V. Co. KG

Dienstag, 4. Juni

09:15	RMAN ist einfach	Johannes Ahrends, Carajan DB GmbH
10:15	Oracle Client Server Kommunikation	Dr. Matthias Mann, Value Transformation Services GmbH
11:15	Applikationsentwicklung: Microservices, REST und JSON aus der Datenbank	Robert Marz, its-people GmbH
12:15	Datentransfer mit Oracle-Tools	Christian Gohmann, Trivadis GmbH
15:15	Patching Oracle: Was muss ich tun?	Martin Bracher, Trivadis AG
16:15	Oracle Cloud Infrastructure: Netzwerk-Einrichtung für DBAs	Robert Marz, its-people GmbH

Codd & ACID

Ein Ausflug in die Theorie und Geschichte der relationalen Datenbanken

Markus Flechtner



BASEL ▪ BERN ▪ BRUGG ▪ DÜSSELDORF ▪ FRANKFURT A.M. ▪ FREIBURG I.BR. ▪ GENÈVE
HAMBURG ▪ KOPENHAGEN ▪ LAUSANNE ▪ MÜNCHEN ▪ STUTTGART ▪ WIEN ▪ ZÜRICH

trivadis
makes IT easier. ■ ■ ■

■ Trivadis – Unsere Mission.

Trivadis makes IT easier:

- Wir unterstützen unsere Kunden massgeblich bei der **intelligenten Nutzung** von Daten im **digitalen Zeitalter**.
- Wir reduzieren **Komplexität** für unsere Kunden durch herausragende **Technologiekompetenz**.
- Wir übernehmen **Kernaufgaben** der bestehenden und zukünftigen IT unserer Kunden.



trivadis
makes IT easier. ■ ■ ■

■ Trivadis – Was uns auszeichnet und unterscheidet.

- Wir verstehen die Business-Prozesse und wirtschaftlichen Herausforderungen unserer Kunden und unterstützen sie durch IT-Beratung und bei der Entwicklung ganzheitlicher IT-Lösungen.
- Unsere selbstentwickelten, bewährten Produkte und Methoden basieren auf dem tiefen Know-how in den Kerntechnologien von Microsoft, Oracle und Open Source. Dies unterscheidet uns von unserem Wettbewerb.



trivadis
makes IT easier. ■ ■ ■

■ Trivadis – Unsere wichtigsten Kennzahlen.



- Gründung: 1994.
- 15 Trivadis Niederlassungen mit über 650 Mitarbeitenden.
- Umsatz CHF 111 Mio. (EUR 96 Mio.).
- Über 250 Service Level Agreements.
- Mehr als 4'000 Trainingsteilnehmer.
- Forschungs- und Entwicklungsbudget: CHF 5.0 Mio.
- Mehr als 1'900 Projekte pro Jahr bei über 800 Kunden.
- Finanziell unabhängig und nachhaltig profitabel.

trivadis
makes IT easier. ■ ■ ■

■ Über mich - Markus Flechtner

- Principal Consultant, Trivadis, Düsseldorf, seit April 2008
- Im Oracle-Umfeld tätig seit den 1990ern:
 - Development (Forms, Reports, PL/SQL)
 - Support
 - Database Administration
- Schwerpunkte
 - Oracle Real Application Clusters
 - Datenbank Upgrade- und Migrationsprojekte
- Kursreferent
 - O-RAC – Oracle Real Application Clusters
 - O-NF-DBA – Oracle Database New Features for the DBA
 - O-MT – Oracle Multitenant
 - PG4ORA – PostgreSQL für Oracle-DBAs



Blog: markusdba.de



@markusdba



DOAG



trivadis
makes IT easier. ■ ■ ■

Technik allein bringt Sie nicht weiter. Man muss wissen, wie man sie richtig nutzt.



■ Eine kurze Geschichte der Datenbanken (1)

1960er	es gibt "Netzwerk-Datenbanken" und "hierarchische Datenbanken"
1970	E.F. Codd veröffentlicht seinen Aufsatz "A relational Model for Large Shared Data Banks"
1970er	Die Entwicklung der relationalen Datenbanksysteme (RDBMS) beginnt
	IBM entwickelt "System R" mit Abfragesprache SEQUEL (" S tructured E nglish Q uery L anguage")
1977	Die Oracle Corporation wird gegründet, zuerst unter dem Namen "Software Development Laboratories" (SDL), 1979 umbenannt in "Relational Software Inc.", Umbenennung in "Oracle" 1983
1979	Oracle V2 erscheint

■ Eine kurze Geschichte der Datenbanken (2)

1983	"ACID" als Akronym für Eigenschaften von Transaktionen wird geprägt
1985	E.F. Codd veröffentlicht die "Codd'schen Regeln"
1998	Die erste "NoSQL"-Datenbank erscheint
2000	Das CAP-Theorem ("Brewers Theorem") wird entwickelt und 2002 bewiesen

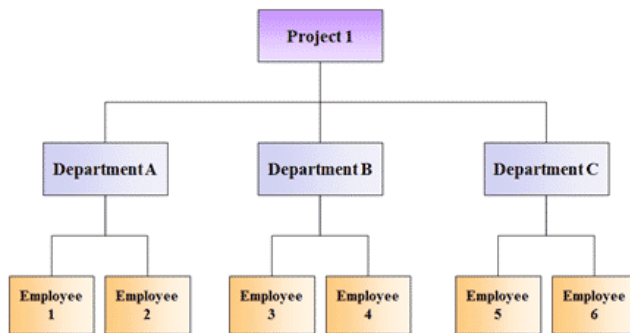
■ Agenda

1. "In the beginning, there was no SQL .."
2. Und dann kam Codd ..
3. ACID – warum es gut ist, wenn die Datenbank sauer ist
4. "At the end, there is NoSQL again" - was ist mit CAP und BASE?

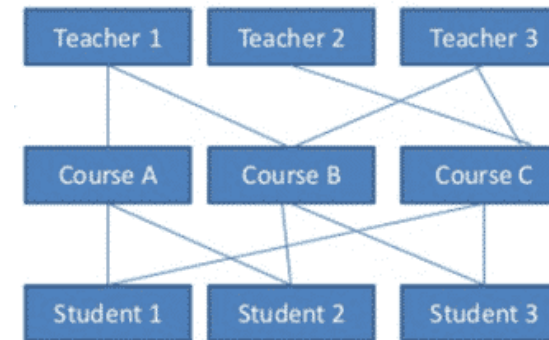
**"In the beginning,
there was no SQL .."**

■ In den 1960ern ...

Hierarchische Datenbanken



Netzwerk-Datenbanken



■ Probleme/Einschränkungen

- Daten können nur eingeschränkt miteinander verknüpft werden
- Datenstrukturen und –beziehungen sind teilweise fest definiert
- Verknüpfung zwischen Daten (Inhalten) und der Art und Weise wie sie physisch abgespeichert werden

**Und dann kam Codd ..
.. und die DB-Welt wurde relational**

■ Edgar Frank „Ted“ Codd

- 1923 - 2003
- Mathematiker
- IBM-Mitarbeiter (1976 IBM Fellow)
- Veröffentlichte 1970 den Aufsatz "A relational Model for Large Shared Data Banks"
- ➔ theoretische Grundlagen der RDBMS
- ➔ "relationale Algebra"

A Relational Model of Data for Large Shared Data Banks

E. F. CODD
IBM Research Laboratory, San Jose, California

Future users of large data banks must be protected from having to know how the data is organized in the machine (the internal representation). A prompting service which supplies such information is not a satisfactory solution. Activities of users at terminals and most application programs should remain unaffected when the internal representation of data is changed and even when some aspects of the external representation are changed. Changes in data representation will often be needed as a result of changes in query, update, and report traffic and natural growth in the types of stored information.

Existing noninferential, formatted data systems provide users with tree-structured files or slightly more general network models of the data. In Section 1, inadequacies of these models are discussed. A model based on n -ary relations, a normal form for data base relations, and the concept of a universal data sublanguage are introduced. In Section 2, certain operations on relations (other than logical inference) are discussed and applied to the problems of redundancy and consistency in the user's model.

KEY WORDS AND PHRASES: data bank, data base, data structure, data organization, hierarchies of data, networks of data, relations, derivability, redundancy, consistency, composition, join, retrieval language, predicate calculus, security, data integrity
CR CATEGORIES: 3.70, 3.73, 3.75, 4.20, 4.22, 4.29

The relational view (or model) of data described in Section 1 appears to be superior in several respects to the graph or network model [3, 4] presently in vogue for non-inferential systems. It provides a means of describing data with its natural structure only—that is, without superimposing any additional structure for machine representation purposes. Accordingly, it provides a basis for a high level data language which will yield maximal independence between programs on the one hand and machine representation and organization of data on the other.

A further advantage of the relational view is that it forms a sound basis for treating derivability, redundancy, and consistency of relations—these are discussed in Section 2. The network model, on the other hand, has spawned a number of confusions, not the least of which is mistaking the derivation of connections for the derivation of relations (see remarks in Section 2 on the "connection trap").

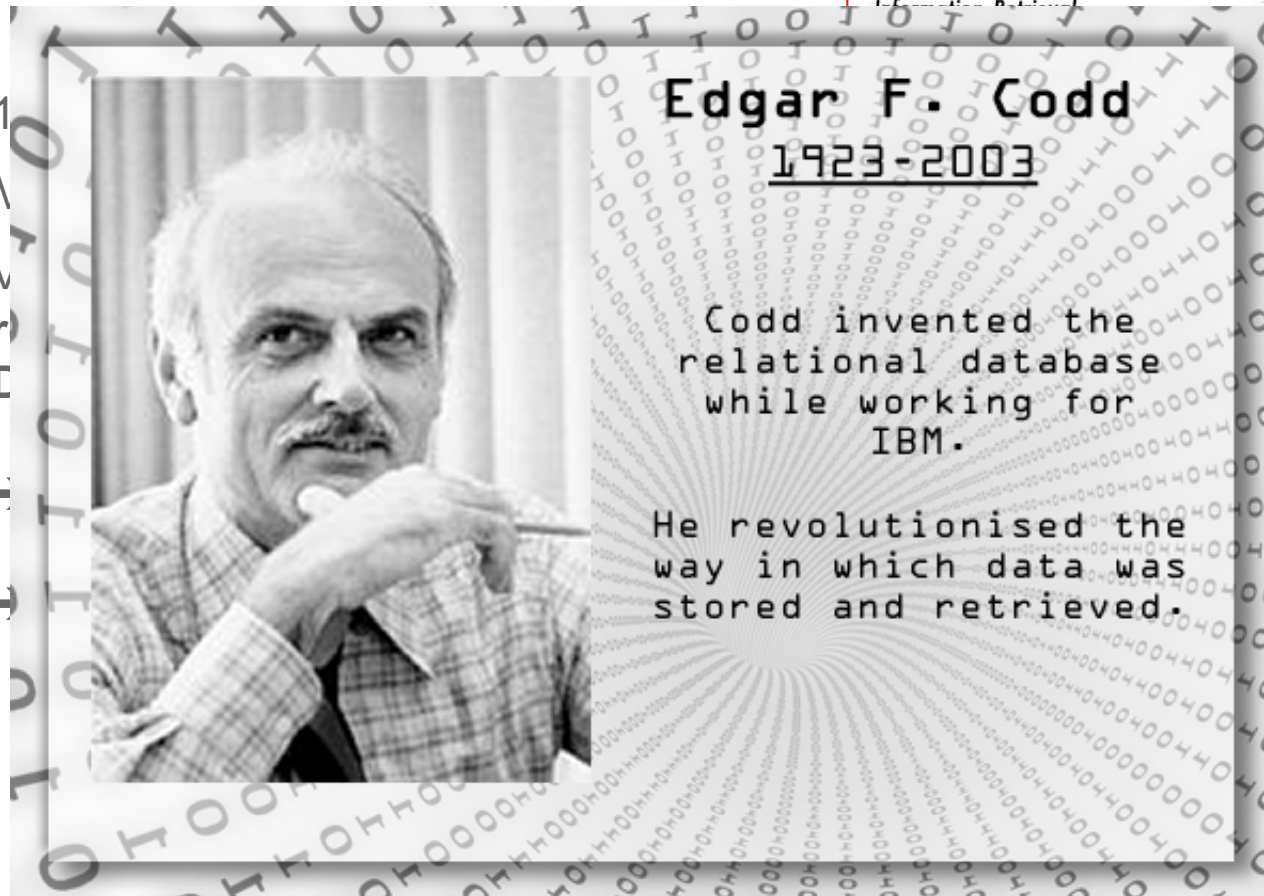
Finally, the relational view permits a clearer evaluation of the scope and logical limitations of present formatted data systems, and also the relative merits (from a logical standpoint) of competing representations of data within a single system. Examples of this clearer perspective are cited in various parts of this paper. Implementations of systems to support the relational model are not discussed.

1.2. DATA DEPENDENCIES IN PRESENT SYSTEMS

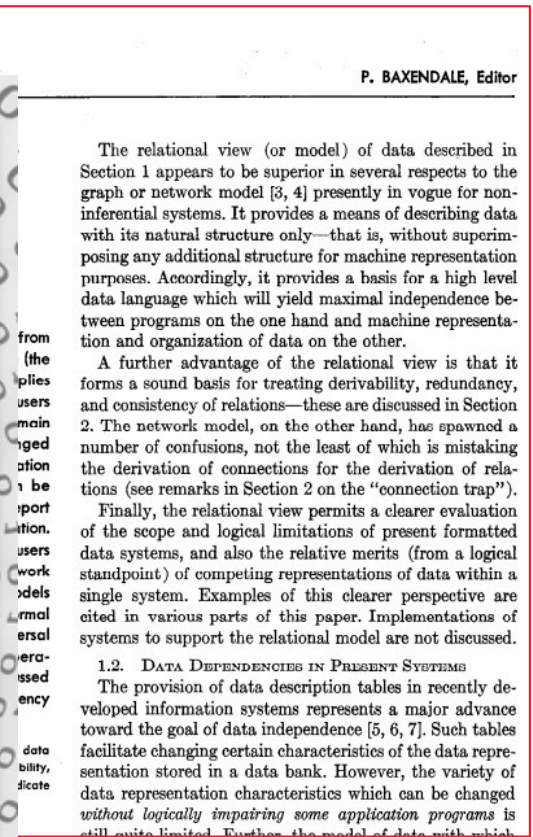
The provision of data description tables in recently developed information systems represents a major advance toward the goal of data independence [5, 6, 7]. Such tables facilitate changing certain characteristics of the data representation stored in a data bank. However, the variety of data representation characteristics which can be changed without logically impairing some application programs is still quite limited. Further, the model of data with which

Quelle: <https://www.seas.upenn.edu/~zives/03f/cis550/codd.pdf>

■ Edgar Frank „Ted“ Codd



Quelle: <https://missicteacher.wordpress.com/ocr-computing/edgar-f-codd/>

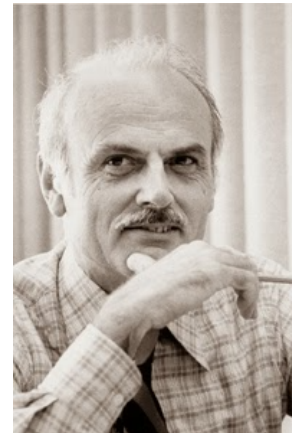


www.cs.cmu.edu/~zives/03f/cis550/codd.pdf

trivadis
makes IT easier. ■ ■ ■

■ Codd'sche Regeln (1)

- Veröffentlicht 1985
- Im Original 12 Regeln



Relational Database Model
12 Rules
by
Edgar Frank Ted Codd

© 87android

Bild: <https://exploredatabase.com/2015/02/codds-twelve-rules.html>

- **Regel 0: (ergänzt 1990):**
"For any system that is advertised as, or claimed to be, a relational data base management system, that system must be able to manage data bases entirely through its relational capabilities."
- → Nur ein System, das die folgenden 12 Regeln (vollständig) erfüllt, darf sich "RDBMS" nennen.
- → Es gibt (strenggenommen) kein "RDBMS", dass alle Regeln vollständig erfüllt.

■ Codd'sche Regeln (2)

■ Regel 1:

"All information in a relational data base is represented explicitly at the logical level and in exactly one way — by values in tables. "

■ → Alle Daten sind in Tabellen abgelegt. Das Datenmodell ist normalisiert.

■ Regel 2:

"Each and every datum (atomic value) in a relational data base is guaranteed to be logically accessible by resorting to a combination of table name, primary key value and column name."

■ → Ein Wert kann über Tabellennamen, Spaltennamen und Primärschlüssel des Datensatzes eindeutig identifiziert werden.

■ Codd'sche Regeln (3)

■ Regel 3:

"Null values (distinct from the empty character string or a string of blank characters and distinct from zero or any other number) are supported in fully relational DBMS for representing missing information and inapplicable information in a systematic way, independent of data type."

■ → NULL-Werte werden innerhalb eines RDBMS einheitlich verarbeitet

■ → Definierte Behandlung des Undefinierten

■ **Achtung:** wie NULL-Werte innerhalb eines RDBMS verarbeitet werden, kann sich je nach Implementierung in Feinheiten unterscheiden

```
SQL> select 'TEST' || null from dual;  
'TES  
----  
TEST
```



Oracle behandelt NULL-
Werte wie Leer-Strings →
Verstoß gegen Regel 3

```
postgres=# select 'TEST' || null;  
?column?  
-----  
  
(1 row)
```

■ Codd'sche Regeln (4)

■ Regel 4:

"The data base description is represented at the logical level in the same way as ordinary data, so that authorized users can apply the same relational language to its interrogation as they apply to the regular data."

- → Auch die Metadaten (Informationen über die Anwendungstabellen) sind in Tabellen gespeichert → **Data Dictionary**

■ Codd'sche Regeln (5)

■ Regel 5:

"A relational system may support several languages and various modes of terminal use (for example, the fill-in-the-blanks mode). However, there must be at least one language whose statements are expressible, per some well-defined syntax, as character strings and that is comprehensive in supporting all of the following items:

Data definition, View definition, Data manipulation, Integrity Constraints, Authorization, Transaction Management"

■ ➔ Es gibt (mindestens) eine "Sprache" für ein RDBMS mit den Komponenten DDL, DML und DCL.

■ ➔ Codd definiert SQL nicht, sondern nur die Anforderungen an eine RDBMS-Sprache. SQL orientiert sich an den Codd'schen Regeln und der relationalen Algebra

■ Codd'sche Regeln (6)

■ Regel 6:

All views that are theoretically updatable are also updatable by the system.

■ → Ein RDBMS muss "updateable views" ermöglichen

■ Regel 7:

The capability of handling a base relation or a derived relation as a single operand applies not only to the retrieval of data but also to the insertion, update and deletion of data.

■ → Ein RDBMS muss mengenorientierte Operationen nicht nur für Abfragen (SELECT), sondern auch für Änderungen (INSERT, UPDATE, DELETE) erlauben.

■ → SQL ist eine mengenorientierte Sprache, Einzelsatzverarbeitung mit SQL ist (meist/fast immer) nicht erforderlich

■ Codd'sche Regeln (7)

■ Regel 8:

Application programs and terminal activities remain logically unimpaired whenever any changes are made in either storage representations or access methods.

- → Für den Benutzer ist es nicht wichtig, wo die Daten abgespeichert sind und wie das DBMS darauf zugreift; das DBMS muss dafür sorgen, dass die richtigen Daten "gefunden" werden.

■ Regel 9:

Application programs and terminal activities remain logically unimpaired when information-preserving changes of any kind that theoretically permit unimpairment are made to the base tables.

- → Abfragen müssen nicht geändert werden, wenn sich die Art, wie die Daten gespeichert werden (Nicht-partitioniert, partitioniert, komprimiert, IOT, ...) ändert.

■ Codd'sche Regeln (8)

■ Regel 10:

Integrity constraints specific to a particular relational data base must be definable in the relational data sublanguage and storable in the catalog, not in the application programs.

■ → man kann (und sollte!) Constraints in der Datenbank definieren (und sie werden von der Datenbank validiert) und nicht in der Applikation!

■ → die Datenbank ist nicht eine dumme Persistenz-Schicht sondern sie kann viel mehr!

■ Regel 11:

A relational DBMS has distribution independence.

■ → RDBMS müssen auch verteilte Datenbanken ermöglichen / verteilte Abfragen erlauben, ohne dass die Anwendung dafür geändert werden muss.

■ Codd'sche Regeln (9)

■ Regel 12:

If a relational system has a low-level (single-record-at-a-time) language, that low level cannot be used to subvert or bypass the integrity rules and constraints expressed in the higher level relational language (multiple-records-at-a-time).

- → man kann die in der Datenbank definierten referentiellen Integritätsregeln mit Mitteln der Datenbank nicht umgehen!

ACID – warum es gut ist, wenn die Datenbank sauer ist

■ Was ist "ACID"?

- Anforderungen an Transaktionen in einem Datenbank-Management-System (DBMS)
- Was ist eine Transaktion?.. **Eine logische zusammenhängende Operation, die aus mehreren Befehlen bestehen kann**
- "ACID" steht für
 - **Atomicity**
 - **Consistency**
 - **Isolation**
 - **Durability**
- Seit den 1970ern "unbewusst" bei vielen DBMS angewendet
- Zuerst als "ACID" formuliert 1983 von Andreas Reuter and Theo Härder

■ A = Atomicity (Atomarität)

- Eine Transaktion wird entweder ganz oder gar nicht ausgeführt ("either all-or-nothing")

```
SQL> UPDATE dept SET loc='SEATTLE' WHERE loc='CHICAGO';  
SQL> UPDATE emp SET ename='KONG' WHERE ename='KING';  
SQL> COMMIT;
```

DEPTNO	DNAME	LOC
30	SALES	SEATTLE

EMPNO	JOB	ENAME
7839	PRESIDENT	KONG



DEPTNO	DNAME	LOC
30	SALES	CHICAGO

EMPNO	JOB	ENAME
7839	PRESIDENT	KING



DEPTNO	DNAME	LOC
30	SALES	CHICAGO

EMPNO	JOB	ENAME
7839	PRESIDENT	KONG



DEPTNO	DNAME	LOC
30	SALES	SEATTLE

EMPNO	JOB	ENAME
7839	PRESIDENT	KING



Bei der Ausführung hat es einen Fehler gegeben, **kein Datensatz wurde geändert**. **DB-theoretisch OK**, praktisch eher nicht

■ C = Consistency (Konsistenz)

- **Nach einer Transaktion sind die Daten logisch konsistent.**
- → die in der Datenbank definierten Integritätsregeln werden eingehalten
 - NOT NULL Constraints
 - PRIMARY Keys
 - UNIQUE Keys
 - FOREIGN Keys
 - CHECK Constraints
- → eine Transaktion, die Integritätsregeln verletzt, schlägt fehl

■ I = Isolation (1)

■ Transaktionen dürfen sich nicht gegenseitig beeinflussen

- Man sieht das, was zu Beginn einer Transaktion COMMITed war."
- Ein Datensatz darf nicht gleichzeitig von 2 Benutzern geändert werden.
- Lesekonsistenz

■ Was kann passieren?

- Dirty Reads
- Lost Updates
- Non-Repeatable Reads
- Phantom Reads

■ I = Isolation (2)

■ Transaction Isolation Level

- ~~– Read Uncommitted~~
- Read Committed (Default bei Oracle)
- Repeatable Read
- Serializable

■ I = Isolation (3)

■ Multi-Version-Concurrency-Control (MVCC)

- Sicherstellen, dass eine Transaktion, das sieht, was zu Beginn der Transaktion COMMITed war
- → **Locking** (zwei Transaktionen können nicht gleichzeitig die gleichen Daten ändern)
- → Bei Änderungen (durch andere Transaktionen) wird der alte Stand gespeichert
- → Oracle: **UNDO**

■ D = Durability (Dauerhaftigkeit)

- Auch Systemfehler verändern die Daten nicht.
- "Commit complete" → Die Daten sind sicher auf Platte gespeichert.
- Oracle gewährleistet "Durability" durch die Redolog-Dateien.

■ AICD - Zusammengefasst

- **Atomicity:** All or nothing
- **Consistency:** Rules kept
- **Isolation:** No partials seen
- **Durability:** Doesn't disappear

Quelle: <https://hub.packtpub.com/postgresqls-transaction-model/>

**"At the end, there is NoSQL again"
- was ist mit CAP und BASE?**

■ NoSQL („Not only SQL“) – Datenbanken

- Ende der 1990er
- Zuerst "No SQL" = "kein SQL", später "Not only SQL, nicht nur SQL"
- Key-Value-Datenbanken
- Graph-Datenbanken
- Document-Datenbanken
- ..
- Für Anwendungsfälle, bei denen RDBMS nicht erforderlich oder zu langsam/aufwändig sind.

■ CAP-Theorem (Brewers Theorem) – 2000/2002

- In einem verteilten System ist es nicht möglich, die Eigenschaften Konsistenz (Consistency), Verfügbarkeit (Availability) und Ausfalltoleranz (Partition Tolerancy) zu garantieren

- **Consistency:**

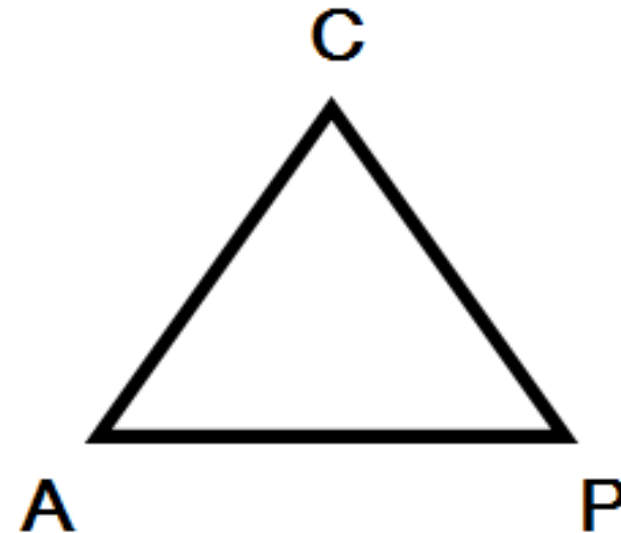
In einem verteilten System sind alle Replikate der Daten identisch.

- **Availability:**

Das System ist immer verfügbar, alle Anfragen werden beantwortet.

- **(Netzwerk-)Partitionstoleranz:**

Das System arbeitet weiter, auch wenn ein Knoten ausfällt.



Quelle: Von Tobias.trelle - Eigenes Werk, CC BY-SA 3.0,
<https://commons.wikimedia.org/w/index.php?curid=16302982>

■ BASE

- Konsistenzmodell für verteilte Systeme

- **Basically Available, Soft state, Eventual consistency**

 - Konstruiertes Akronym, "BASE" als Gegensatz zu "ACID"

- → "Eventuell zeigen alle Abfragen den letzten Stand eines Datensatzes"

- "Langfristig" sind alle Knoten konsistent

■ ACID vs. BASE (1)

■ "Consistency" (ACID) != "Consistency" (BASE)

ACID	BASE
"starke" Konsistenz	"schwache" Konsistenz
Schwerpunkt "Commit"	Priorität "Performance"
komplexer	einfacher
exakt	möglicherweise nicht immer exakt
aufwändig & langsam	einfacher & schneller

■ ACID vs. BASE (2) – oder: RDBMS vs. NoSQL

- Beide Modelle haben ihre Berechtigung.
- Der Anwendungsfall entscheidet.

**Google darf gerne BASE für seine Suchergebnisse nutzen.
Mir ist egal, ob es 999.963 oder 998.763 Treffer für meine Suche gibt.**

**Aber meine Bank macht das für die Anzeige
meines Kontostandes und für meine Überweisungen hoffentlich nicht.**

Weitere Informationen



- Edgar F. Codd - "A Relational Model of Data for Large Shared Data Banks." – z.B. <http://www.seas.upenn.edu/~zives/03f/cis550/codd.pdf>
- Codd's 12 Regeln – z.B. <http://www.insidesql.org/blogs/frankkalis/2004/07/13/codds-12-regeln> oder https://en.wikipedia.org/wiki/Codd's_12_rules
- Tom Kyte on Transaction Isolation Levels: <https://blogs.oracle.com/oraclemagazine/on-transaction-isolation-levels>
- MOS-Note "Master Note: Oracle Transaction Management (Local) Overview (Doc ID 1506115.1)"
- How does a relational database work - <http://coding-geek.com/how-databases-work/>
- Brewer's CAP Theorem - <http://www.julianbrowne.com/article/brewers-cap-theorem>

Fragen & Antworten

Markus Flechtner
Principal Consultant

Tel. +49 211 5866 64725

Markus.Flechtner@Trivadis.com

 @markusdba <https://markusdba.de>

Heute, 17:15:

Stairway to Heaven - Die Zukunft des DBA
Was begegnet mir auf dem Weg in die Cloud?

Pinwand für Fragen und Anmerkungen im Foyer
=> Antworten gibt es in der Open-Mic-Session



Vortrag zum Download verfügbar unter <https://www.slideshare.net/markusflechtner>

Bitte die Vortragsbewertung nicht vergessen – Danke!

trivadis
makes IT easier. ■ ■ ■