# COIN-Datenmodelle

*Peter Sonntag, VIZSON e.K.; Greifswalder Str. 33a; 10405 Berlin; Germany*

*Berlin, 18. Februar 2015*

## 1. Inhaltsverzeichnis

## 2. Beschreibung

**COIN-Datenmodelle** (von engl. **CO**lumn **IN**dependent, **spaltenunabhängig**) ordnen Informationen Angaben in weiteren Spalten zu und erweitern somit das Key/Value-Modell um eine informationsbasierte Dimension. COIN-Datenmodelle unterscheiden sich damit von zeilen- oder spaltenorientierten Datenmodellen durch die Zuordnung von Informationen. COIN-Datenmodelle leiten tupelbildende Tabellenschemata aus den schlüsselgebenden Spalten der informationshaltenden Tabellen und Zuordnungen ab und nicht durch die tatsächlich vorhandenen Tabellen.

Zeilenorientierte Datenmodelle ordnen Informationen einer Zeile und einer Spalte und ggf. fremdschlüsselbezogenen Zeilen aus anderen Tabellen zu. Spaltenorientierte Datenmodelle ordnen Informationen einer Zeile und ggf. fremdschlüsselbezogenen Zeilen aus anderen Tabellen zu.

COIN-Datenmodelle hingegen ordnen Informationen Angaben zu, die in weiteren Spalten untergebracht sind.

Beispiel:

| **Schlüssel** | | **Wert** |
|---|---|---|
| **Datensatzkennung** | **Spaltenkennung** | **Wert** |
| 1 | Landname | USA |
| 1 | Kontinent | Nordamerika |
| 2 | Landname | Kanada |
| 2 | Kontinent | Nordamerika |

*Tabelle 1: Tabelle eines spaltenunabhängigen Datenmodells*

| **Primärschlüssel** | **Landname** | **Kontinent** |
|---|---|---|
| 1 | USA | Nordamerika |
| 2 | Kanada | Nordamerika |

*Tabelle 2: Tabelle eines zeilenorientierten Datenmodells*

| **Primärschlüssel** | **Landname** |
|---|---|
| 1 | USA |
| 2 | Kanada |

*Tabelle 3a*

| **Primärschlüssel** | **Kontinente** | **Fremdschlüssel Länder** |
|---|---|---|
| 1 | Nordamerika | 1,2 |

*Tabelle 3b*

*Tabellen 3a und 3b: Tabellen „Ländernamen“ und „Länderkontinente“ eines spaltenorientierten Datenmodells.*

## 3. Vorteile

Vorteile des spaltenunabhängigen Datenmodells ergeben sich aus seiner Flexibilität:

### 3.1. Normalisierte Tabellenstruktur

Da sämtliche informationshaltenden Tabellen dieselbe Struktur besitzen, können Abfragen generisch formuliert werden. Besonders bei komplexen Datenstrukturen kann daher z.B. die Entwicklungszeit von Anwendungen gesenkt werden.

### 3.2. Unterschiedliche Datenstrukturen in einer Tabelle

Während in zeilenorientierten Datenmodellen für weitere Wertestrukturen neue Tabellen erstellt werden, können in spaltenunabhängigen Tabellenmodellen Werte für unterschiedliche Datenstrukturen in der gleichen Tabelle gehalten werden:

| **Schlüssel** | | **Wert** |
|---|---|---|
| **Datensatzkennung** | **Spaltenkennung** | **Wert** |
| 1 | Landname | USA |
| 1 | Kontinent | **1** |
| 2 | Landname | Deutschland |
| 2 | Kontinent | 2 |
| **1** | Kontinentname | Nordamerika |
| 2 | Kontinentname | Europa |

*Tabelle 4: Werte für unterschiedliche Datenstrukturen in der gleichen Tabelle: Die Datensatzkennung 1 der Spaltenkennung „Kontinentname“ wird als Wert der Datensatzkennung 1 der Spaltenkennung „Kontinent“ abgelegt.*

### 3.3. Schlüsselerweiternde Spalten

Durch die Einführung schlüsselerweiternde Spalten können Datenstrukturen erweitert werden. Z.B. für sprachsensitive Werte:

| **Schlüssel** | | | **Wert** |
|---|---|---|---|
| **Datensatzkennung** | **Sprachkennung** | **Spaltenkennung** | **Wert** |
| 1 | 1 | Landname | Vereinigte Staaten von Amerika |
| 1 | 2 | Landname | United States Of America |
| 1 | 0 | Kontinent | 1 |
| 2 | 1 | Landname | Deutschland |

| | | | |
|---|---|---|---|
| 2 | 2 | Landname | Germany |
| 2 | 0 | Kontinent | 2 |
| 1 | **1** | Kontinentname | Nordamerika |
| 1 | 2 | Kontinentname | North America |
| 2 | 1 | Kontinentname | Europa |
| 2 | 2 | Kontinentname | Europe |
| **1** | 1 | Sprache | Deutsch |
| 1 | 2 | Sprache | German |
| 2 | 1 | Sprache | Englisch |
| 2 | 2 | Sprache | English |

*Tabelle 5: Abbildung sprachsensitiver Informationen: Die Datensatzkennung 1 der Spaltenkennung „Sprache“ wird als Sprachkennung der Datensatzkennung 1 der Spaltenkennung „Kontinentname“ abgelegt.*

Dies gilt analog z.B. auch für die Behandlung von Mehrfachantworten:

| **Schlüssel** | | | **Wert** |
|---|---|---|---|
| **Datensatzkennung** | **Antwortkennung** | **Spaltenkennung** | **Wert** |
| 1 | 0 | Landname | Schweiz |
| 1 | **1** | Sprache | NULL |
| 1 | **2** | Sprache | NULL |
| 1 | **3** | Sprache | NULL |
| 1 | **4** | Sprache | NULL |
| **1** | 0 | Sprachenname | Deutsch |
| **2** | 0 | Sprachenname | Französich |
| **3** | 0 | Sprachenname | Italienisch |
| **4** | 0 | Sprachenname | Rätoromanisch |

*Tabelle 6: Abbildung Mehrfachantworten für eine Spalte: Die Datensatzkennungen 1,2,3,4 der Spaltenkennung Sprachenname werden als Antwortkennung der Datensatzkennung 1 der Spaltenkennung „Sprache“ abgelegt*

### 3.4. Werteerweiternde Spalten

Weitere Spalten können für informationsbasierte Metadaten genutzt werden:

| **Schlüssel** | | | **Wert** | **Metainfo n** |
|---|---|---|---|---|
| **Datensatzkennung** | **Sprachkennung** | **Spaltenkennung** | **Wert** | **Besitzer** |
| 1 | 0 | Kontinent | 1 | 1 |
| 1 | 1 | Landname | Vereinigte Staaten von Amerika | 2 |

| | | | | |
|---|---|---|---|---|
| 1 | 2 | Landname | United States Of America | 2 |

*Tabelle 7: Informationsbasierte Metadaten: Die Spaltenkennung „Kontinent“ der Datensatzkennung 1 ist im Besitz von Besitzer 1, die Spaltenkennung „Landname“ der Datensatzkennung 1 ist im Besitz von Besitzer 2. In diesem Fall ist eine simultane Bearbeitung der Datensatzkennung 1 durch Besitzer 1 und Besitzer 2 möglich, sofern nur die jeweils eigenen Informationen bearbeitet werden.*

Weitere Beispiele wären z.B. die Gültigkeitsdauer einer Information, nach der sie durch eine Routine gelöscht würde. Damit eignet sich ein spaltenunabhängiges Datenmodell beispielsweise sehr gut für ein System, das Informationen selbstständig „vergessen“ können soll.

### 3.5. Beliebige Spaltenattribute

Die Verwaltung von Spaltenattributen auf die gleiche Weise hat den Vorteil, dass beliebig viele Spaltenattribute eingeführt werden können. Dies ist z.B. für eine generische Wertevalidierung vorteilhaft.

| **Schlüssel** | | **Wert** |
|---|---|---|
| **Datensatzkennung** | **Attributkennung des Spaltenattributes** | **Wert** |
| **1** | 1 | Spaltenname |
| 2 | 1 | maximal zulässige Zeichenanzahl |

*Tabelle 8: Spaltenattribute*

| **Schlüssel** | | **Wert** |
|---|---|---|
| **Datensatzkennung** | **Spaltenattributkennung** | **Wert** |
| **1** | **1** | Landname |
| **1** | 2 | 50 |
| 2 | **1** | Kontinentname |
| 2 | 2 | 70 |

*Tabelle 9: Spaltenattributwerte: Die Datensatzkennung 1 der Tabelle „Spaltenattribute“ wird als Spaltenattributkennung der Datensatzkennungen 1 und 2 der Tabelle „Spaltenattributwert“ abgelegt. Daraus wird der „Spaltenname“ der Spaltenattributkennung 1 ( „Landname“) ersichtlich.*

| **Schlüssel** | | **Wert** |
|---|---|---|
| **Datensatzkennung** | **Spaltenkennung** | **Wert** |
| 1 | **1** | USA |
| 1 | 2 | Nordamerika |

*Tabelle 10: Informationen: Die Datensatzkennung 1 der Tabelle „Spaltenattributwerte“ wird als Spaltenkennung der Datensatzkennung 1 der Tabelle „Informationen“ abgelegt. Daraus folgt, dass der Wert den „Landnamen“ ausdrückt und in diesem Fall maximal 50 Zeichen lang sein darf.*

3.6. Implizite Verwaltung von Fremdschlüsselbeziehungen

Im Gegensatz zu zeilenorientierten Modellen ist die individuelle Kenntnis der Fremdschlüsselbeziehungen nicht notwendig, da Fremdschlüsselbeziehungen wiederum als Datensätze geführt werden können. Das bietet diese Vorteile:

1. Der Grad der Inkonsistenz ist steuerbar, indem das gewünschte Verhalten (Löschen des ganzen Datensatzes der Fremdschlüsselbeziehung/Löschen nur der Information im Datensatz der Fremdschlüsselbeziehung/Zulassen einer Inkonsistenz) bei jeder Fremdschlüsselbeziehung mitgeführt werden kann.
2. Falls es sich um eine Verknüpfung zu z.B. Werten für ein Auswahlfeld handelt, können
    a. sowohl die Informationen über Sortierschlüssel und -richtung für jede einzelne Fremdschlüsselbeziehung als auch
    b. die anzuzeigenden Felder der Fremdschlüsseldatensätze und ggf. deren Verbindung mitgeführt werden (z.B. Nachname, Vorname).
3. Gerade bei größeren Systemen führt eine systematische Verwaltung der Fremdschlüsselbeziehungen zu einem System, in dem die Fremdschlüsselbeziehungen sehr leicht nachzuvollziehen sind und sich direkt aus dem System ergeben.

3.7. Balancierung

Die Informationen einer Spalte können sich in jeder beliebigen informationshaltenden Tabelle befinden. Das führt zu einer Möglichkeit der automatischen Optimierung einer Balancierung der Datenmengen. Falls die Zeilenanzahl einer Tabelle zu nachteiligen Einflüssen führt, können die Werte für eine oder mehrere Spalten in eine neue Tabelle übertragen werden ohne das Gesamtsystem zu beeinflussen. Dabei ist es unerheblich, ob nur die Werte für eine oder für mehrere Spalten in einer Tabelle vorhanden sind.

3.8. Reduktion auf vorhandene Werte

Auf einem spaltenunabhängigen Datenmodell basierende Systeme können so ausgeprägt werden, dass sie eine Entscheidung zulassen, sich auf die Behandlung von inhaltlich vorhandenen Informationen zu beschränken und leere Informationen (z.B. „“, NULL) nicht zu berücksichtigen. Das kann für das lesende System einen Performancevorteil bedeuten, erfordert allerdings auch eine Identifizierungsschicht für die einzelnen Datenbankhandlungen und eine Prüfungsschicht bei der Nutzung von aus dem Datenbanksystem an das abfragende System zurückgelieferten Ergebnissen.

### 3.9. Informationsbasierte Historisierung

Eine informationsbasierte Historisierung ist aufgrund der identischen Tabellenstruktur der informationshaltenden Tabellen für sämtliche Datenhandlungen in parallel geführten identischen Tabellen mit zusätzlichen Spalten für die Identifikation der Historisierungsschicht realisierbar. Bei dieser Art der Historisierung entstehen signifikant weniger Daten als bei einer Protokollierung von Datenbankabfragen. Zusätzlich lassen auf spaltenunabhängigen Datenmodellen beruhende Systeme eine gezielte Abfrage von ausgewählten Historisierungsdaten unter Nutzung der gleichen systematischen Abfrage wie zur Abfrage von Produktivdaten zu.

## 4. Nachteile

### 4.1. Tabellennamen

Da die Namen der informationshaltenden Tabellen für die Abfrage nicht notwendig sind, müssen diese vor jeder Abfrage aus informationshaltenden Tabellen identifiziert werden. In großen auf zeilenorientierten Datenmodellen sowie in auf spaltenorientierten Datenmodellen basierenden Systemen ist dies allerdings ebenfalls notwendig.

### 4.2. Datentypen

Eine Nutzung mehrerer Datentypen verbreitert die informationshaltenden Tabellen um eine Spalte pro Datentyp und erzeugt n-1 leere Zellen pro Information. Das stellt allerdings erst dann einen Nachteil dar, wenn die Nutzung von leeren Zellen den Speicherbedarf signifikant erhöht.

### 4.3. Datenbankhandlungen

Ähnlich wie bei spaltenorientierten Datenmodellen, sind je nach Ausprägung unterschiedliche Datenbankhandlungen notwendig, um die in zeilenorientierten Datenmodellen üblichen Handlungen (Einfügen, Abfragen, Aktualisieren, Löschen) zu realisieren. Bei einer Ausprägung, die auf eine Verwaltung von leeren Informationen (z.B. „“, NULL) verzichtet, sind weitere Anpassungen notwendig. In diesem Fall kann eine Aktualisierung z.B. die Löschung einer Zeile bedeuten. In dieser Ausprägung muss die Ergebnismenge zusätzlich auf die Existenz einer Information geprüft werden, denn es entstehen dynamische Tupel, die aus unterschiedlich vielen Informationen eines aus zeilenorientierter Sicht definierten Datensatzes bestehen können.

### 4.4. Datensatzbildung in Applikationen

Bei einem Betrieb der Modelle auf einem zeilenbasierten System (z.B. SQL) müssen Applikationen ggf. wesentlich mehr „Datensätze“ der Rückgaberessource einer Abfrage durchlaufen, um die benötigte Zuordnung der Informationen aufzubauen, da jede Information in einer eigenen Zeile untergebracht ist. Diesem Nachteil könnte begegnet werden, in dem die

Abfragesyntax des zeilenbasierten Systems erweitert würde um die Möglichkeit, eine abhängige Ressource zurückzugeben. Modellhaft sei hier eine entsprechende Erweiterung erwähnt:

Da sämtliche strukturellen Abhängigkeiten für jede Information bereits in der Tabelle vorhanden sind und somit in die Ergebnismenge aufgenommen werden können, könnte zumindest bei auf zeilenorientierter Arbeitsweise genutzten Tabellensystem (z.B. SQL-Systemen) die Abhängigkeiten der Abfrageergebnisse bereits in der Abfrage formulieren werden:

Die Abfrage
SELECT table.information AS ARRAY (col1,coln) FROM table

Würde diese Ressource zurückliefern:
$result$_{col1,coln}$=Wert

Hierzu könnte auch der Index eines neuen Typs genutzt werden:
CREATE DEPENDENCY INDEX index1 ON table (col1,coln)
SELECT table.information USING index1 FROM table

Diese Vorgehensweise würde signifikanten Performancevorteil bieten, weil eine das Abfrageergebnis nutzende Anwendung auf ein Durchlaufen der Ergebnismenge verzichten könnte.

## 5. Datenkonsistenz und Normalform

Spaltenunabhängige Datenmodelle erlauben die Bildung von unregelmäßigen Tabellen für jede denkbare Datenstruktur und ermöglichen deren Anpassung. Die Normalform dieser Tabellen ist nicht vorgeschrieben.

Die in Umsetzungen von auf spaltenunabhängigen Datenmodellen basierenden Systemen genutzte Tabellenstruktur stellt hingegen ausschließlich Werte/Schlüsselpaare dar, wobei die Schlüsselelemente aus hierarchischen Abhängigkeiten bestehen können. Diese Simplizität lässt eine Interpretation des Primärschlüssels über mehrere Spalten zu. Somit bildet eine Umsetzung von auf spaltenunabhängigen Datenmodellen basierenden Datenmodellen zunächst untereinander abhängigkeitsfreie Werte ab und bleibt in sich konsistent.

## 6. Weblinks


- Daniel J. Abadi, Samuel R. Madden, Nabil Hachem: Column Stores vs. Row-Stores: How Different Are They Really?: http://db.csail.mit.edu/projects/cstore/abadi-sigmod08.pdf
- Mike Stonebraker, Daniel Abadi, Adam Batkin, Xuedong Chen, Mitch Cherniack, Miguel Ferreira, Edmond Lau, Amerson Lin, Sam Madden, Elizabeth O'Neil, Pat O'Neil, Alex Rasin, Nga Tran and Stan Zdonik. VLDB, pages 553-564, 2005. : C-Store: A Column Oriented DBMS: http://db.csail.mit.edu/projects/cstore/vldb.pdf

- Nikita Ivanov: Columnar vs. Key-Value Storage Models: http://www.gridgain.com/columnar-vs-key-value-storage-models/
- Peter Sonntag: COIN data model: https://vizson.de/coin_datamodel.php

## 7. Literaturangaben


- Edgar F. Codd: The Relational Model for Database Management. Version 2. Addison-Wesley, Reading u. a. 1990, ISBN 0-201-14192-2
- Clive Finkelstein (1992): "Information Engineering: Strategic Systems Development". Sydney: Addison-Wesley.
- James Martin and Clive Finkelstein (1981): Information engineering. Technical Report (2 volumes), Savant Institute, Carnforth, Lancs, UK.
- James Martin (1989): Information engineering. (3 volumes), Prentice-Hall Inc.
- Paulraj Ponniah (2007): "Data Modeling Fundamentals: a practical guide for IT professionals", John Wiley & Sons, Inc. Hoboken, New Jersey, ISBN 0-471-79049-4


## 8. Vervielfältigungshinweis