Datenbanken: Datenkompression

Dr. Matthias Uflacker, Stefan Klauck

2. Mai 2018

HPI Hasso Plattner Institut
IT Systems Engineering | Universität Potsdam

Vorlesungsinhalte/-aufbau

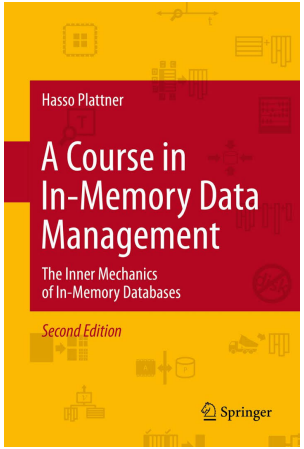
- Phase 1
  - Einführung zu Unternehmensanwendungen (2 Vorlesungen)
  - **Grundlagen von spaltenorientierten Hauptspeicherdatenbanken (4 Vorlesungen)**
  - Wöchentliche Übungsblätter
- Phase 2
  - Grundlagen des IT-gestützten Rechnungswesens und Planung (3 Vorlesungen)
  - Programmiermodelle für Unternehmensanwendungen (1 Vorlesung)
  - Zwei praktische Programmierübungen
- Klausur

Überblick

- Motivation
- Dictionary-Kodierung
  - Allgemeines
  - Komprimierung für den Attributvektor
  - Komprimierung für das Dictionary
- Zusammenfassung
- Übungsblatt 3

Literatur und Empfehlung

- Hasso Plattner „A Course in In-Memory Data Management”



- Daniel Abadi et al. „Integrating Compression and Execution in Column-Oriented Database Systems“ (2006)

  http://db.csail.mit.edu/projects/cstore/abadisigmod06.pdf

Motivation

- Speicherzugriff ist oft der entscheidende Performanzfaktor bei Datenbanksystemen (CPU vs. Hauptspeichergeschwindigkeit)

Idee: Tausche geringere Speicherzugriffskosten gegen Dekomprimierungsmehraufwand

- Zusätzlich ist Hauptspeicher trotz wachsender Kapazität eine begrenzte (und im Vergleich zu Festplatten kostspielige) Ressource

Klassifizierung

- **Leichtgewichtige** (vs. schwergewichtige) Kompression
  - Datenbanksystem soll Daten trotz Kompression ohne größeren Aufwand verarbeiten können (idealer Weise funktionieren Operationen auf komprimierten Daten)
  - Bei Festplatten-IO kommen auch schwergewichtige Verfahren in Frage (da der Geschwindigkeitsunterschied von CPU und Festplatten-IO größer ist)
- **Verlustfreie** (vs. verlustbehaftete) Kompression
  - Reduktion der Bitanzahl bei gleichem Informationsgehalt

Dictionary-Kodierung

- Dictionary(Wörterbuch)-Kodierung ist eine verlustfreie Kompressionsmethode und Grundlage für weitere Kompressionsverfahren

- Idee: Kodiere jeden unterschiedlichen Wert eines Vektors (mit potenziell sehr großen und häufig vorkommenden Werten) mit einer unterschiedlichen Wert-ID (die klein ist)

Dictionary-Kodierung
Beispiel

- **Zu komprimierender Vektor** V = [Australia, USA, Germany, USA]

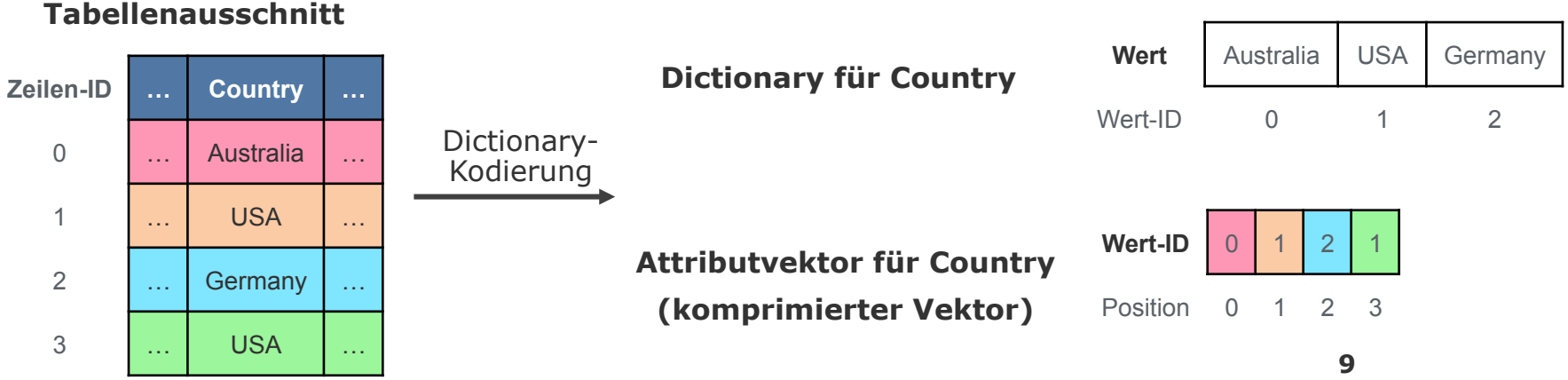
Dictionary-Kodierung 

- Kodierung k (**Dictionary**):
  - Australia -> 0
  - USA -> 1
  - Germany -> 2

- **Kodierter/komprimierter Vektor** k(V) = [0, 1, 2, 1]

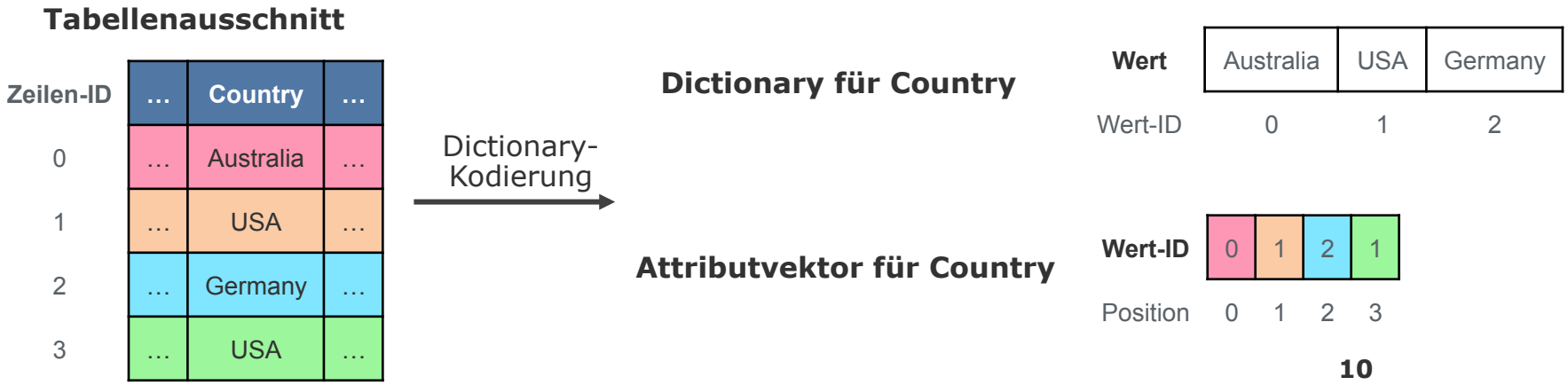
Dictionary-Kodierung
Umsetzung in Datenbanken

- **Direkter Zugriff auf komprimierte Werte** (vs. größte Kompression)
  - Feste Breite von Wert-IDs (minimale Anzahl von Bits vs. Padding)
- Datenstruktur für das Dictionary (Kodierung k)
  - Hier: Vektor, wobei das Offset (Position) die Wert-ID implizit bestimmt



Dictionary-Kodierung
Umsetzung in Datenbanken

- **Dictionary** speichert alle verschiedenen Werte mit einer impliziten Wert-ID
- **Attributvektor (komprimierter Vektor)** speichert Wert-IDs für alle Einträge der Spalte



Dictionary-Kodierung
Kompressionsrate der Spalte Country

- Weltbevölkerung: 8 Milliarden Tupel

Zeilen-ID	First Name	Last Name	Country	Year of Birth
0	Paul	Smith	Australia	1986
1	Lena	Jones	USA	1990
2	Hanna	Schulze	Germany	1942
3	Hanna	Schulze	USA	2000
…	…	…	…	…

Dictionary-Kodierung
Kompressionsrate der Spalte Country

- Weltbevölkerung: 8 Milliarden Tupel
- Annahme: 200 verschiedene Länder, alle werden mit 50 Bytes gespeichert
  - Unkomprimiert: $8 * 10^9 * 50\ B = 400 * 10^9\ B = 400\ GB$
  - Dictionary-Größe: $200 * 50\ B = 10.000\ B = 10\ KB = 0{,}00001\ GB$
  - Benötigte Bits pro Wert-ID: $ceil(\log_2(200))\ b = 8\ b$
  - Größe des Attributvektors: $8 * 10^9 * 8b = 8 * 10^9\ B = 8\ GB$
  - Kompressionsrate = unkomprimierte Größe / komprimierte Größe
    $= 400\ GB / (8{,}00001\ GB) \approx 50$

Dictionary-Kodierung
Daten anfragen

- Suche alle Personen aus den USA
  - Suche die Wert-ID für den gesuchten Wert (USA)
  - Suche die gefundene Wert-ID (1) im Attributvektor

Tabellenausschnitt

Zeilen-ID	…	Country	…
0	…	Australia	…
1	…	USA	…
2	…	Germany	…
3	…	USA	…

Dictionary-Kodierung

Dictionary für Country

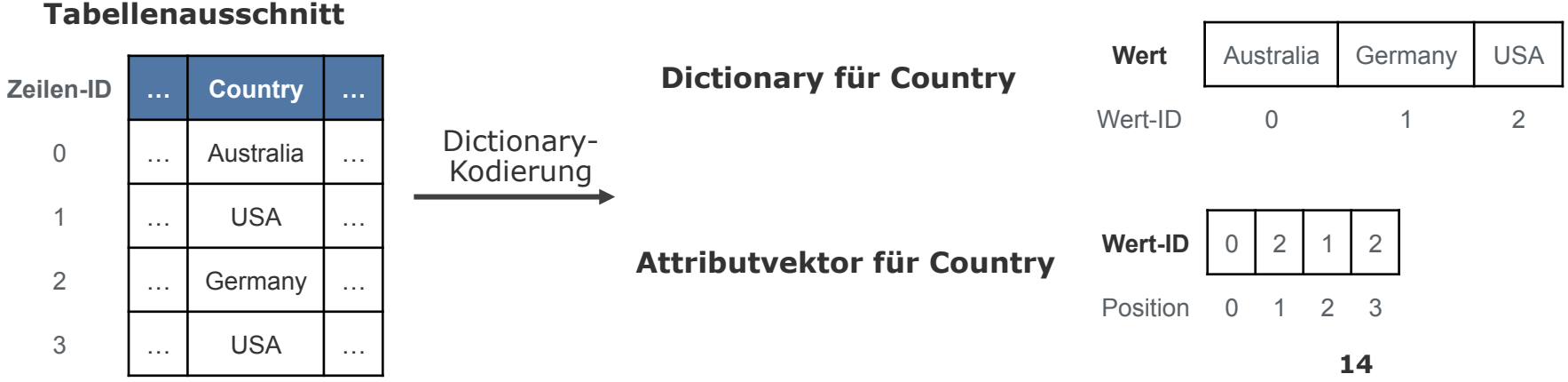
Wert	Australia	USA	Germany
Wert-ID	0	1	2

Attributvektor für Country

Wert-ID	0	1	2	1
Position	0	1	2	3

Dictionary-Kodierung
Sortiertes Dictionary: Vorteile

- Dictionary-Einträge sind nach Wert sortiert
  - Suche der Wert-ID im Dictionary hat Komplexität O(log(n)) statt O(n)
  - Bereichsabfragen (range queries) beschleunigen
  - Dictionary kann besser komprimiert werden



Dictionary-Kodierung
Nachteile

- Sortiertes Dictionary
  - „Umsortierung“ für jeden neuen Wert, der nicht ans Ende des Dictionary gehört (vergleichsweise günstig)
  - Attributvektor aktualisierten (teuer, da dieser (meist) größer ist)

- Allgemein
  - Zusätzliche Indirektion für Dictionary-Lookup (z.B. bei der Materialisierung von Anfrage(zwischen)ergebnissen aus Positionslisten oder beim Einfügen neuer Werte)

Komprimierung für den Attributvektor und für das Dictionary

- Komprimierung für den Attributvektor

  (diese Verfahren funktionieren (z.T. mit Anpassungen) auch für unkomprimierte Vektoren)

  - Run-Length-Encoding (Lauflängenkodierung)
  - Präfix-Kodierung
  - Cluster-Kodierung
  - Indirekte Kodierung

- Komprimierung für das Dictionary
  - Delta-Kodierung

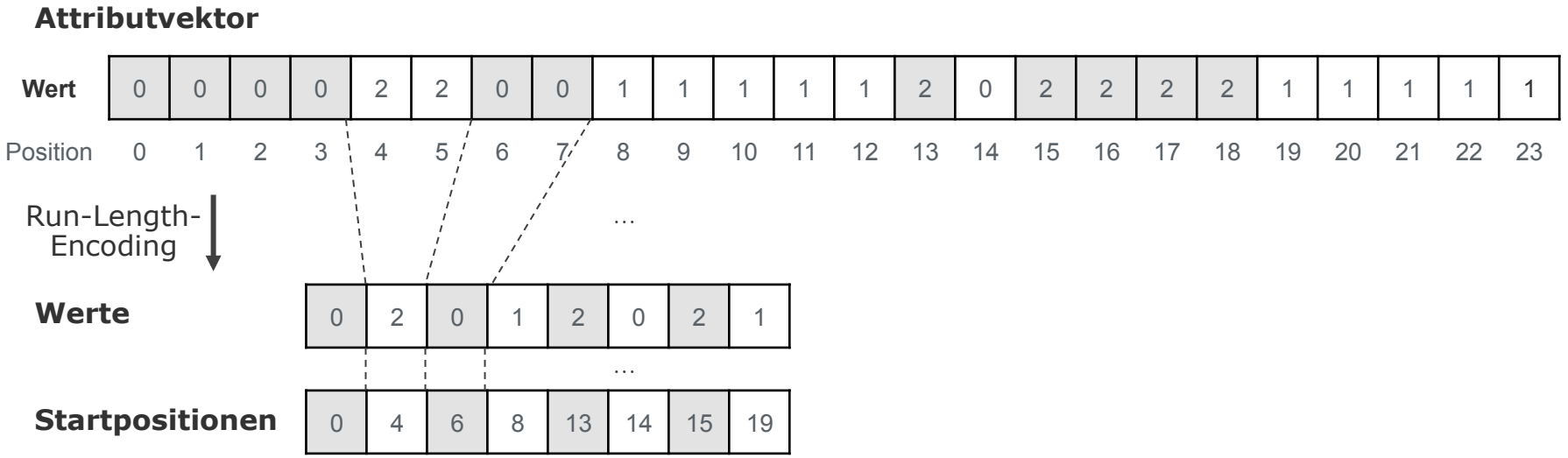
Komprimierung für den Attributvektor
Run-Length-Encoding (Lauflängenkodierung)

- Komprimierungsvervahren für Sequenz von sich wiederholenden Werten
- Grundidee: Speicherung von Wert und dessen Anzahl
- Verschiedene Verfahren/Implementierungen

- Beispiel im Datenbankkontext:
  - Wiederholung von Werten als Tupel (Wert, Startposition, Wiederholungen)
  - Optimierung: Für Vektoren kann die Anzahl der Wiederholungen aus Startpositionen berechnet werden (andersrum kann die Startposition auch aus allen vorherigen Wiederholungen berechnet werden)

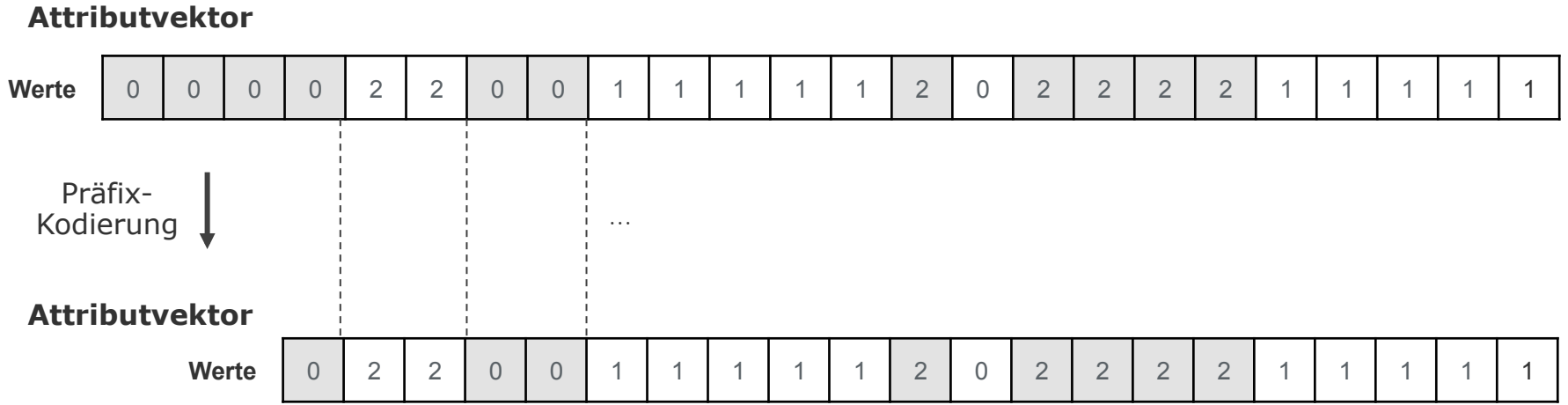
- Gut für sortierte Spalten (viele Wiederholungen)

Komprimierung für den Attributvektor
Run-Length-Encoding - Beispiel



Komprimierung für den Attributvektor
Präfix-Kodierung

- „Run-Length-Encoding des ersten Wertes“
- Direkter Zugriff auf Attributvektor

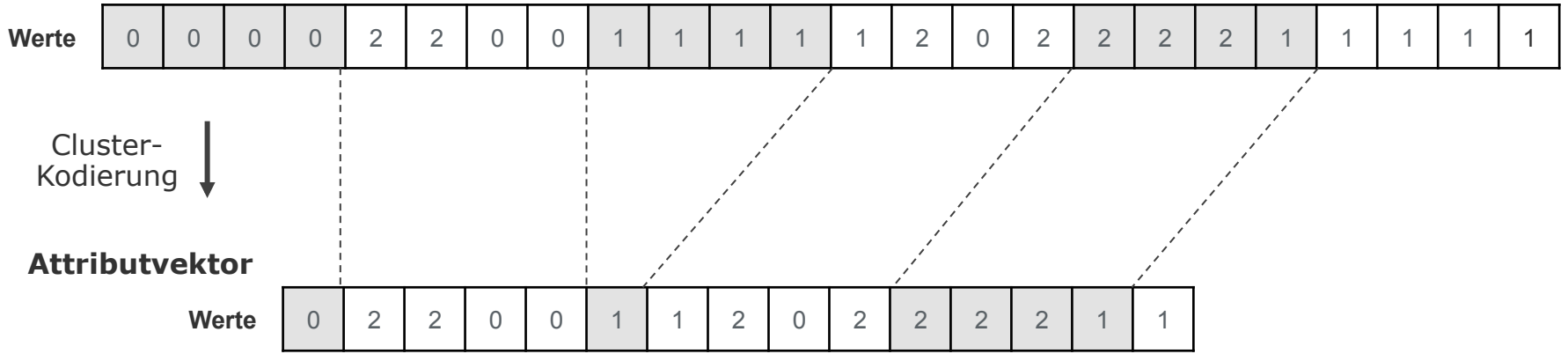


Anzahl des ersten Wertes: 4

Komprimierung für den Attributvektor
Cluster-Kodierung

- Vektor ist in Blöcke/Cluster fester Größe zerlegt
- Falls ein Cluster nur gleiche Werte beinhaltet, wird das Cluster durch diesen Wert kodiert
- Ein Bitvektor kennzeichnet, welche Cluster kodiert sind

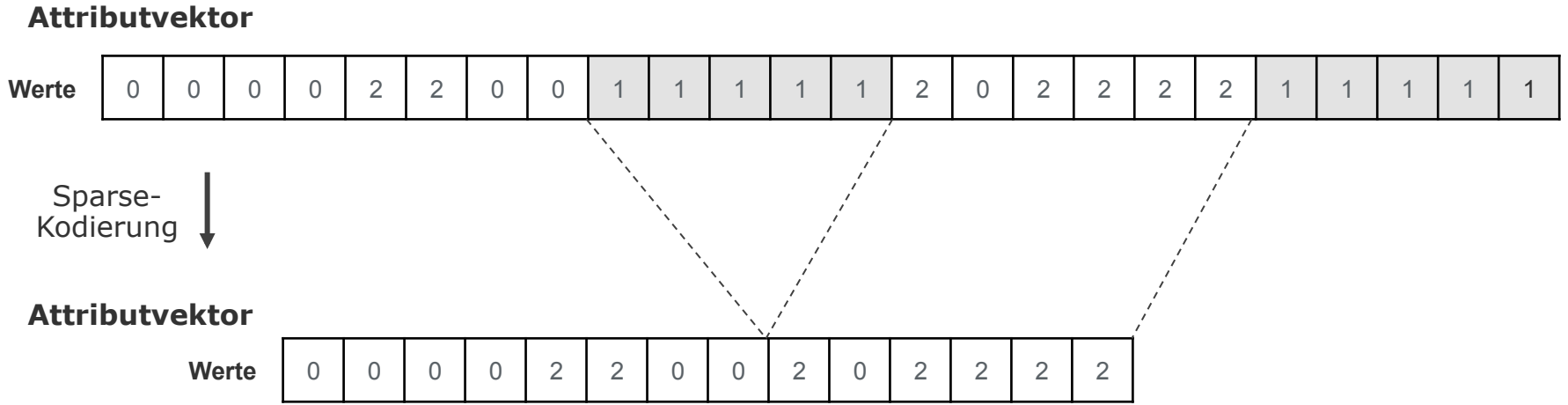
Attributvektor (Blockgröße: 4)



Bitvektor **10 10 01** (1 falls Cluster-kodierter Block)

Komprimierung für den Attributvektor
Sparse-Kodierung

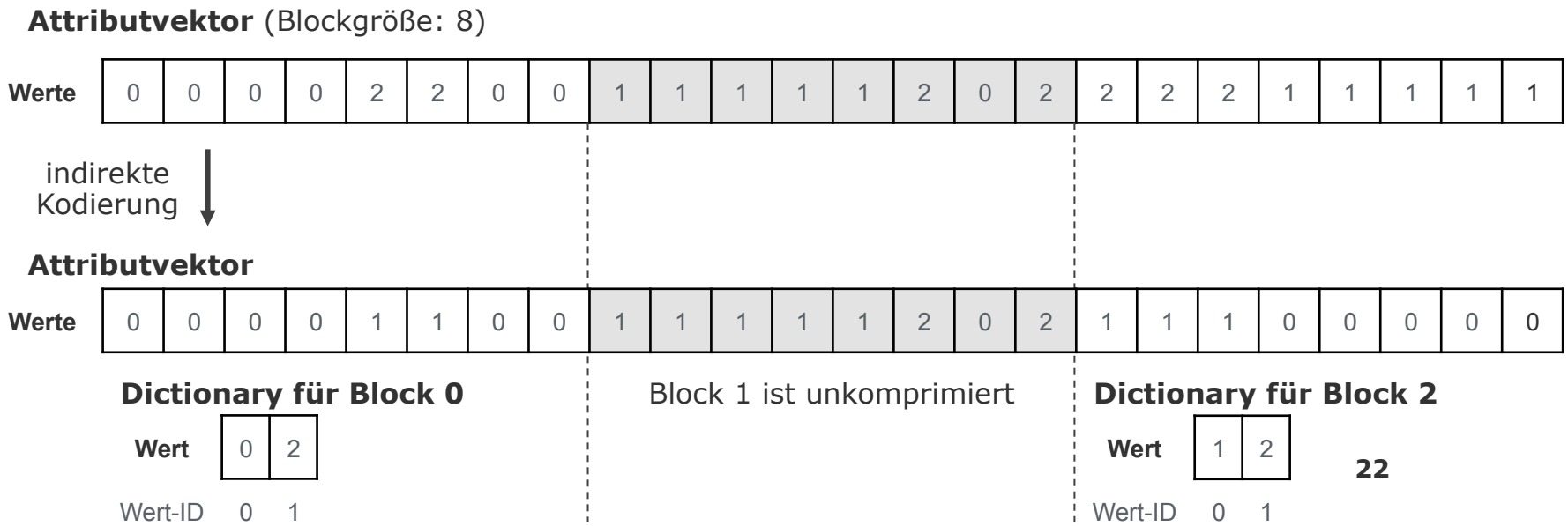
- Entferne den Wert, der am häufigsten im Vektor vorkommt
- Ein Bitvektor kennzeichnet, an welchen Positionen ein Wert entfernt wurde



Bitvektor **00000000 11111000 00011111** (1 falls Wert entfernt)

Komprimierung des Attributvektors
Indirekte Kodierung

- Vektor ist in Blöcke fester Größe zerlegt
- Falls ein Block wenige verschiede Wert enthält, wird er über ein zusätzliches Dictionary kodiert; die Wert-IDs werden dann mit weniger Bits dargestellt



Komprimierung für das Dictionary
Delta-Kodierung

- Vektor **vom Typ String** ist in Blöcke mit einer festen Anzahl von Werten zerlegt
- Blöcke werden einzeln komprimiert und speichern für jeden Wert:
  - Länge des Präfix, welchen der Wert mit dem **unmittelbaren Vorgänger** gemeinsam hat
  - Anzahl der zusätzlichen Zeichen ohne den gemeinsamen Präfix
  - Zusätzlichen Zeichen

Dictionary

Werte	Bahamas	Bahrain	Bangladesh	Barbados	Belarus	Belgium

Delta-Kodierung ↓

Werte	7	Bahamas	3	4	rain	2	8	ngladesh	2	6	rbados	1	6	elarus	3	4	gium

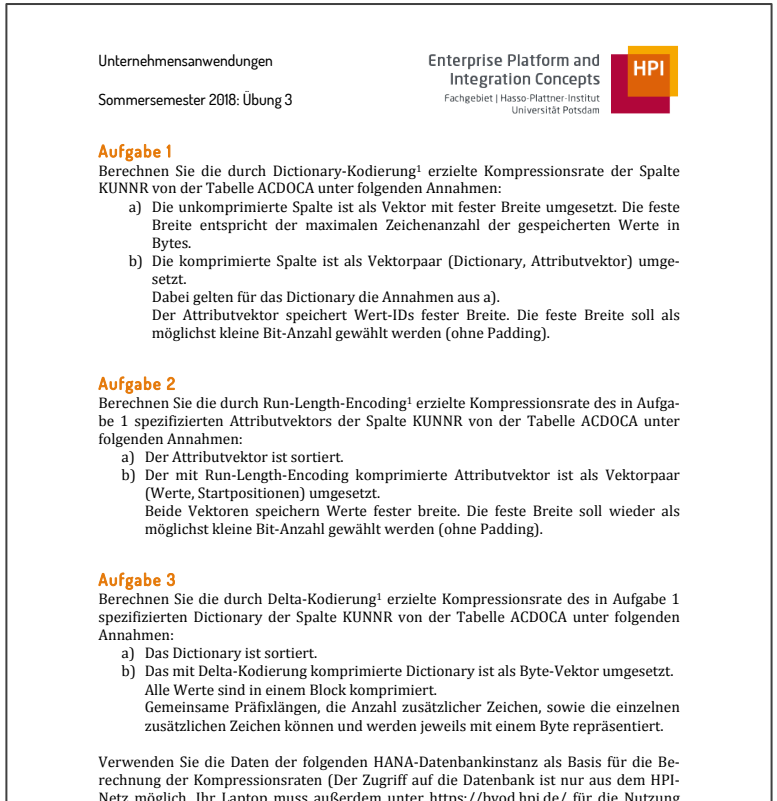
(Vereinfachte Sicht:
Länge muss auch hier gespeichert werden)

Zusammenfassung

- Neben der Reduktion des Speicherbedarfs ist Komprimierung eine Möglichkeit dem Speichzugriffsengpass heutiger Computer entgegenzuwirken
- Besondere wichtig für Hauptspeicherdatenbanken sind Komprimierungsverfahren, die ein schnelles Suchen und direkten Datenzugriff (ohne teure Dekompression) erlauben
- Spaltenorientierte Datenbanken ermöglichen (im Allgemeinen) höhere Kompressionsraten, da sich Werte eines Attributs (meist) ähnlicher sind als die eines Tupels
- Viele Kompressionsverfahren sind erst für sortierte Listen effizient, aber Tabellen können nur nach einer Spalte oder kaskadierend sortiert werden

Unternehmensanwendungen
Übungsblatt 3

Hasso Plattner Institut



Unternehmensanwendungen

Sommersemester 2018: Übung 3

Enterprise Platform and Integration Concepts
Fachgebiet | Hasso-Plattner-Institut Universität Potsdam

Aufgabe 1

Berechnen Sie die durch Dictionary-Kodierung[1] erzielte Kompressionsrate der Spalte KUNNR von der Tabelle ACDOCA unter folgenden Annahmen:

a) Die unkomprimierte Spalte ist als Vektor mit fester Breite umgesetzt. Die feste Breite entspricht der maximalen Zeichenanzahl der gespeicherten Werte in Bytes.
b) Die komprimierte Spalte ist als Vektorpaar (Dictionary, Attributvektor) umgesetzt.
Dabei gelten für das Dictionary die Annahmen aus a).
Der Attributvektor speichert Wert-IDs fester Breite. Die feste Breite soll als möglichst kleine Bit-Anzahl gewählt werden (ohne Padding).

Aufgabe 2

Berechnen Sie die durch Run-Length-Encoding[1] erzielte Kompressionsrate des in Aufgabe 1 spezifizierten Attributvektors der Spalte KUNNR von der Tabelle ACDOCA unter folgenden Annahmen:

a) Der Attributvektor ist sortiert.
b) Der mit Run-Length-Encoding komprimierte Attributvektor ist als Vektorpaar (Werte, Startpositionen) umgesetzt.
Beide Vektoren speichern Werte fester breite. Die feste Breite soll wieder als möglichst kleine Bit-Anzahl gewählt werden (ohne Padding).

Aufgabe 3

Berechnen Sie die durch Delta-Kodierung[1] erzielte Kompressionsrate des in Aufgabe 1 spezifizierten Dictionary der Spalte KUNNR von der Tabelle ACDOCA unter folgenden Annahmen:

a) Das Dictionary ist sortiert.
b) Das mit Delta-Kodierung komprimierte Dictionary ist als Byte-Vektor umgesetzt.
Alle Werte sind in einem Block komprimiert.
Gemeinsame Präfixlängen, die Anzahl zusätzlicher Zeichen, sowie die einzelnen zusätzlichen Zeichen können und werden jeweils mit einem Byte repräsentiert.

Verwenden Sie die Daten der folgenden HANA-Datenbankinstanz als Basis für die Berechnung der Kompressionsraten (Der Zugriff auf die Datenbank ist nur aus dem HPI-Netz möglich. Ihr Laptop muss außerdem unter https://byod.hpi.de/ für die Nutzung