Hasso Plattner Institut

IT Systems Engineering | Universität Potsdam

CBLOCK: An Automatic Blocking Mechanism for Large-Scale De-duplication Tasks

Cathleen Ramson, Stefan Lehmann

LSDD SS 2013

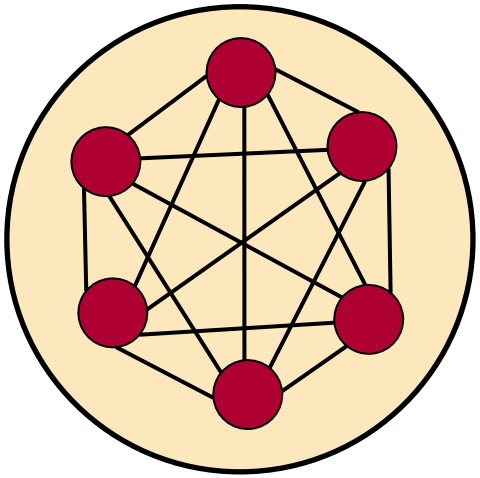
25.04.2013

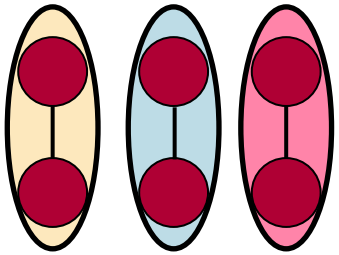
Gliederung

- Motivation
- Ziel
- Algorithmen
- Zusammenfassung
- Bewertung

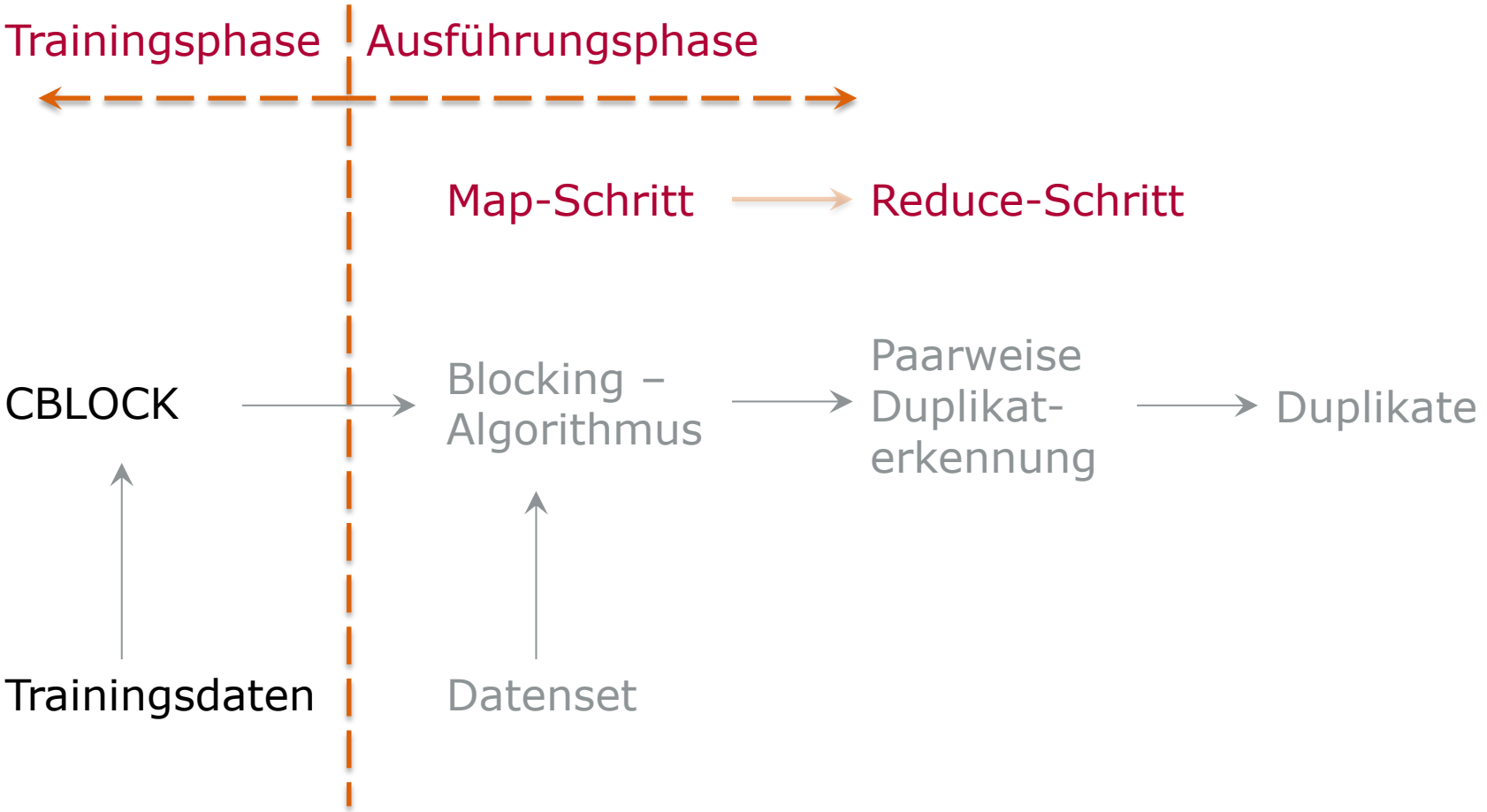
Motivation

- Blocking verringert Komplexität
- Blocking Functions automatisch erstellen
  - Weniger manueller Aufwand
  - Für optimales Load Balancing



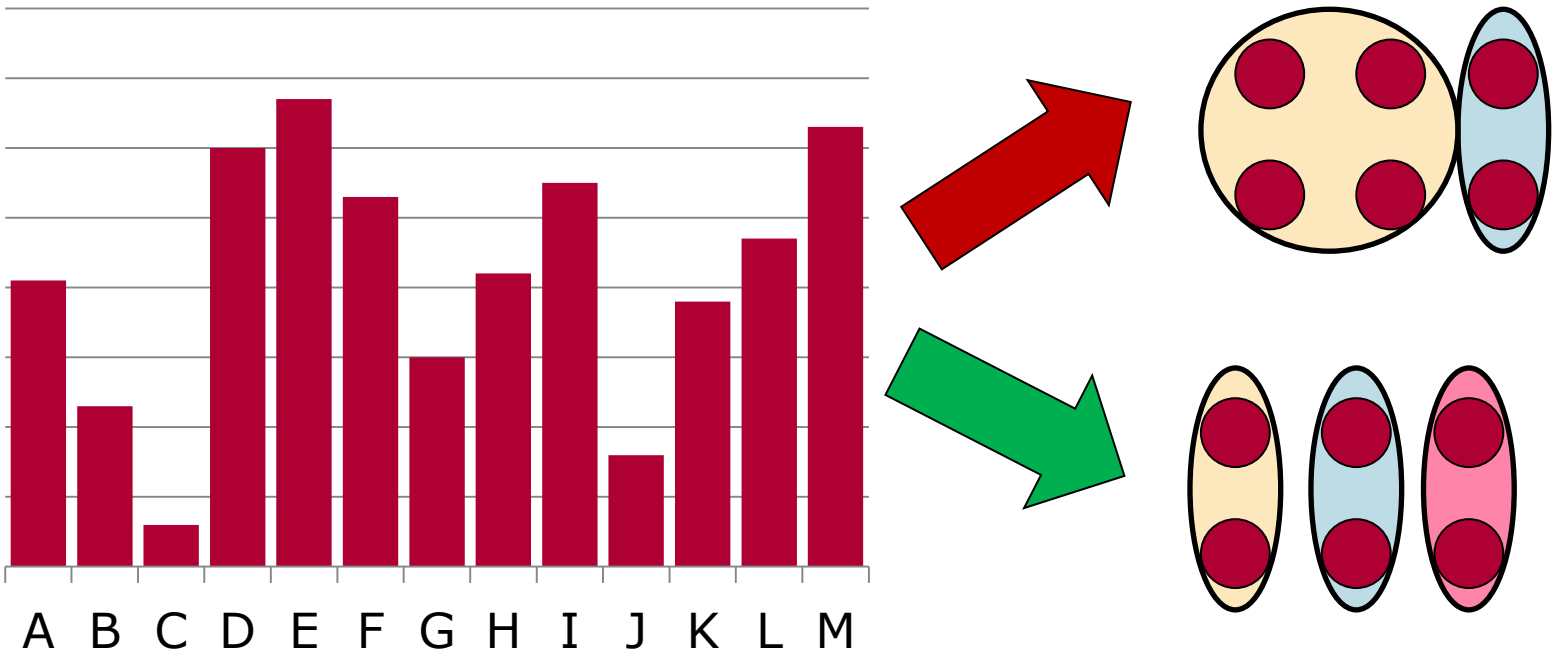


Einordnung


Trainingsphase
Ausführungsphase
Map-Schritt
Reduce-Schritt
CBLOCK
Blocking – Algorithmus
Paarweise Duplikat-erkennung
Duplikate
Trainingsdaten
Datenset

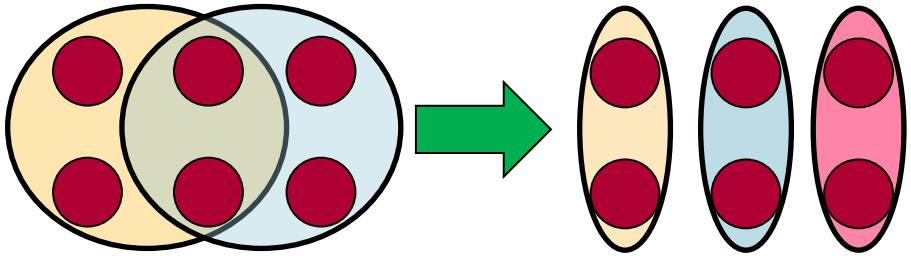
Ziel

- Ein Map-Reduce Durchlauf genügt
- Trotz ungleichmäßiger Datenverteilung einheitliche Blockgröße



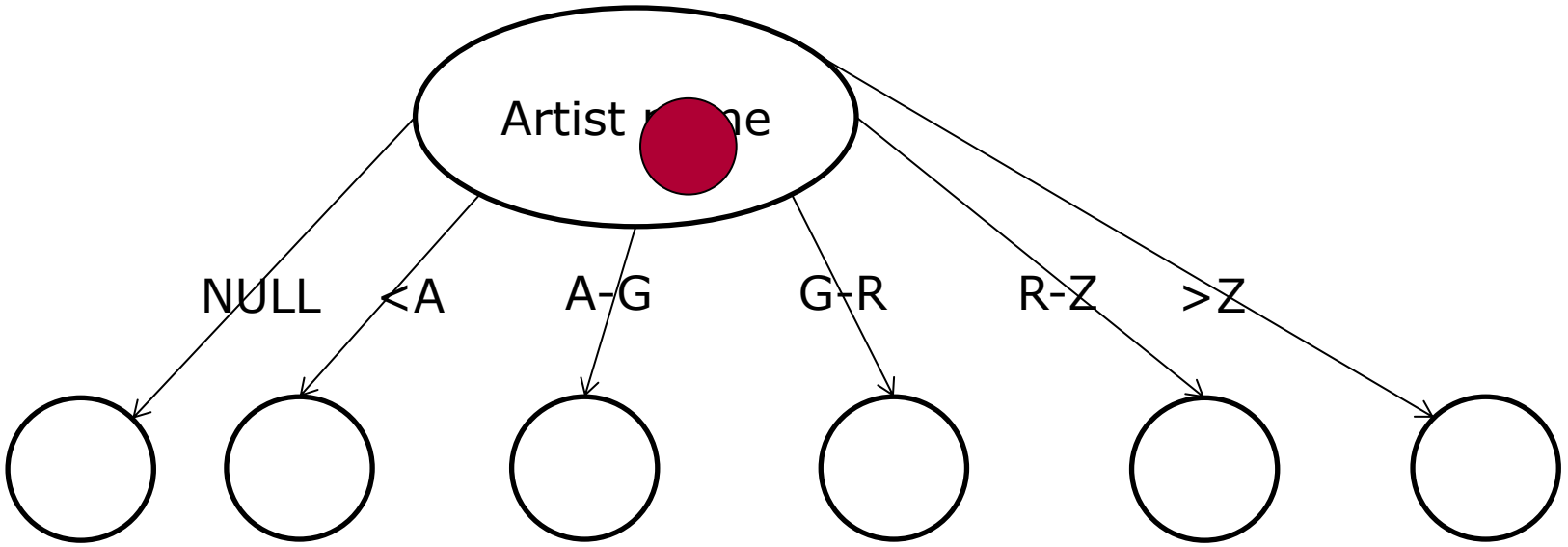
Ziel

- Ein Map-Reduce Durchlauf genügt
- Trotz ungleichmäßiger Datenverteilung einheitliche Blockgröße
- Kein Element in mehreren Blöcken



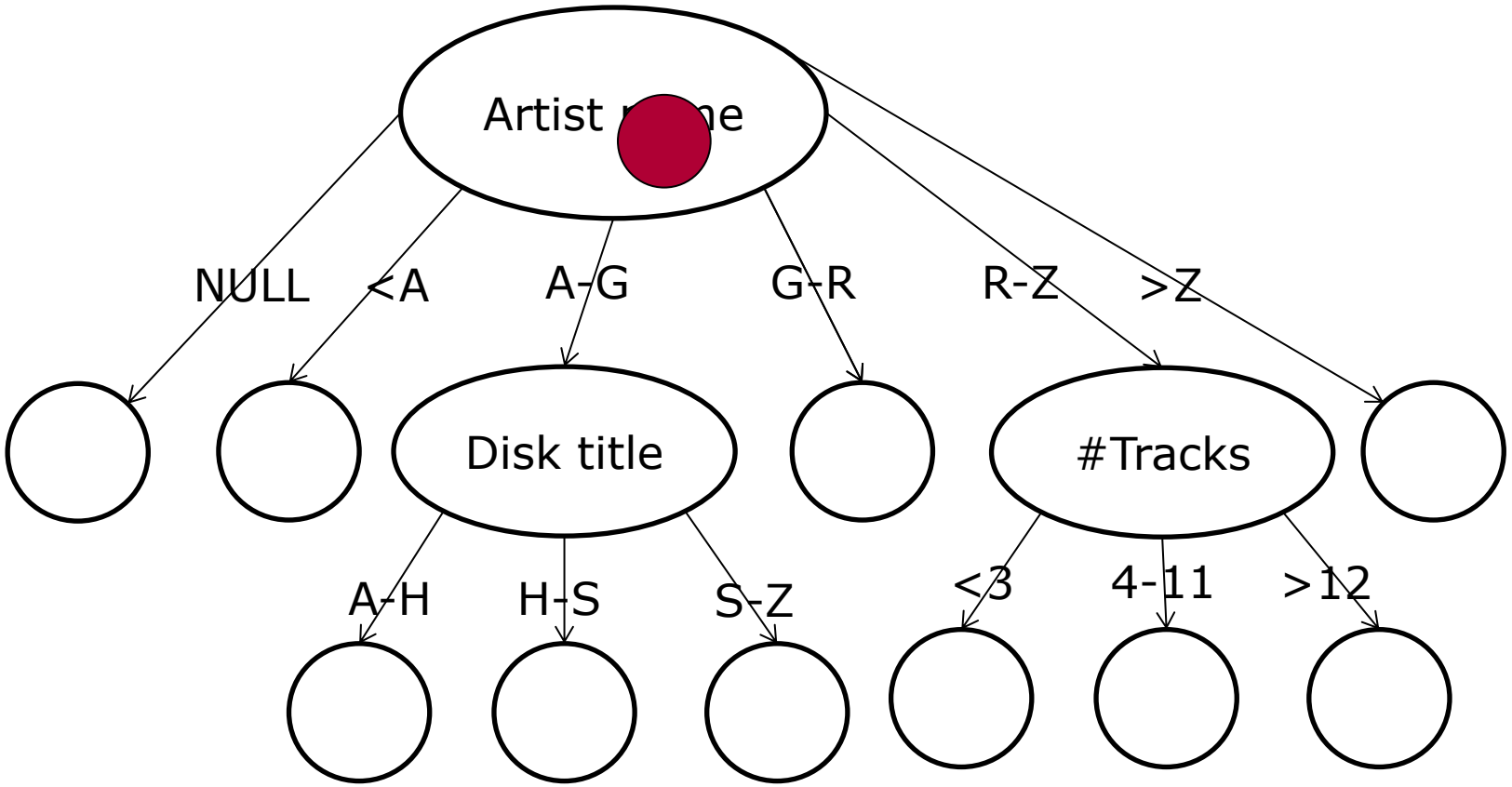
Blocking Tree

- The Rolling Stones; Bridges To Babylon; 13



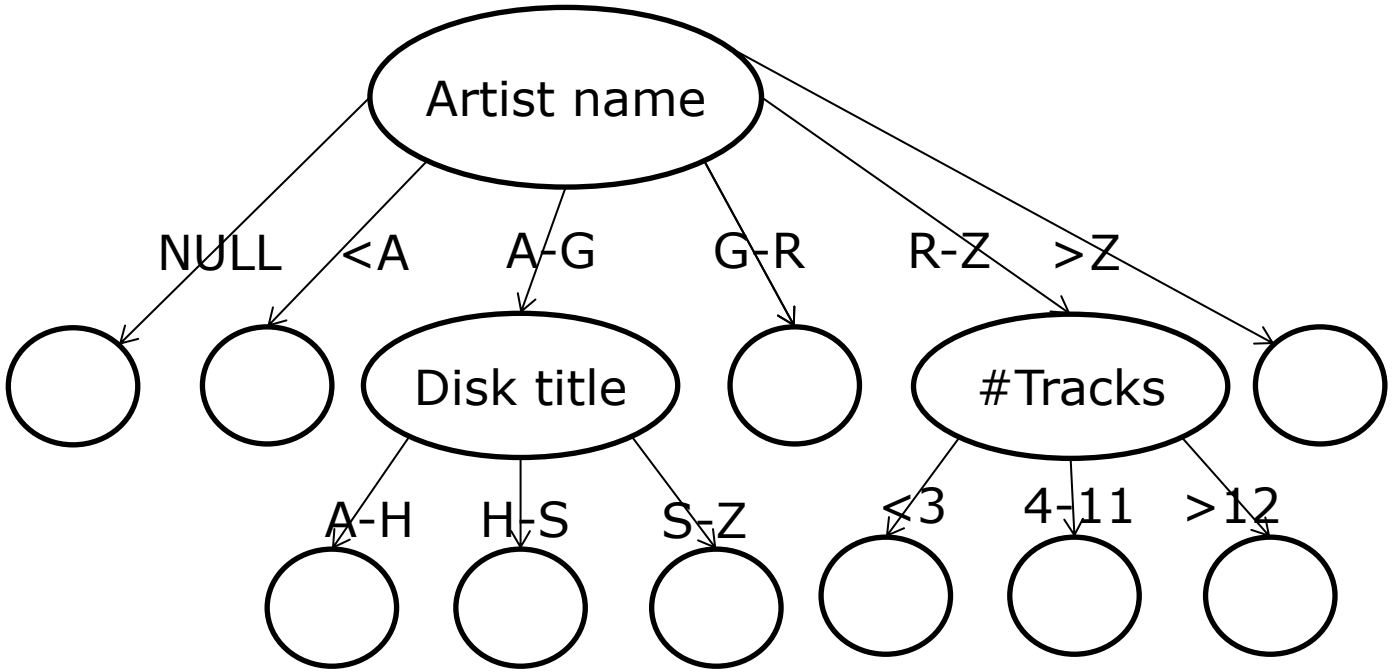
Blocking Tree

- The Rolling Stones; Bridges To Babylon; 13



Erlernen optimaler Bäume

- Voraussetzung
  - Mögliche Hashfunktionen gegeben
  - Trainingsdaten gegeben



Erlernen optimaler Bäume

- Wähle Hashfunktion mit wenigsten getrennten Duplikaten
  - Schätze Größe aller Kindknoten
  - Wiederholung für alle großen Kindknoten

Disk title

Watazumi Doso; Music Of Japan; 21
≠
Watazumi Doso; The Art Of Japanese …; 21

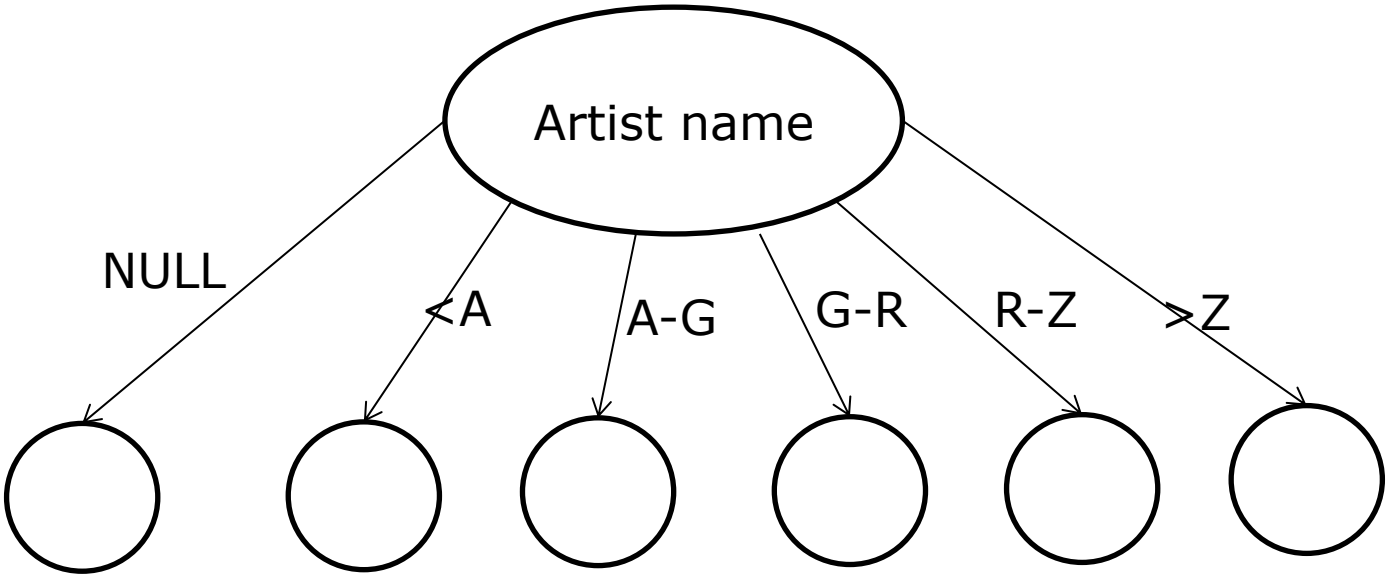
#Tracks

Jennifer Lopez; J. Lo; 15
≠
Jennifer Lopez; J. Lo; 16

Artist name

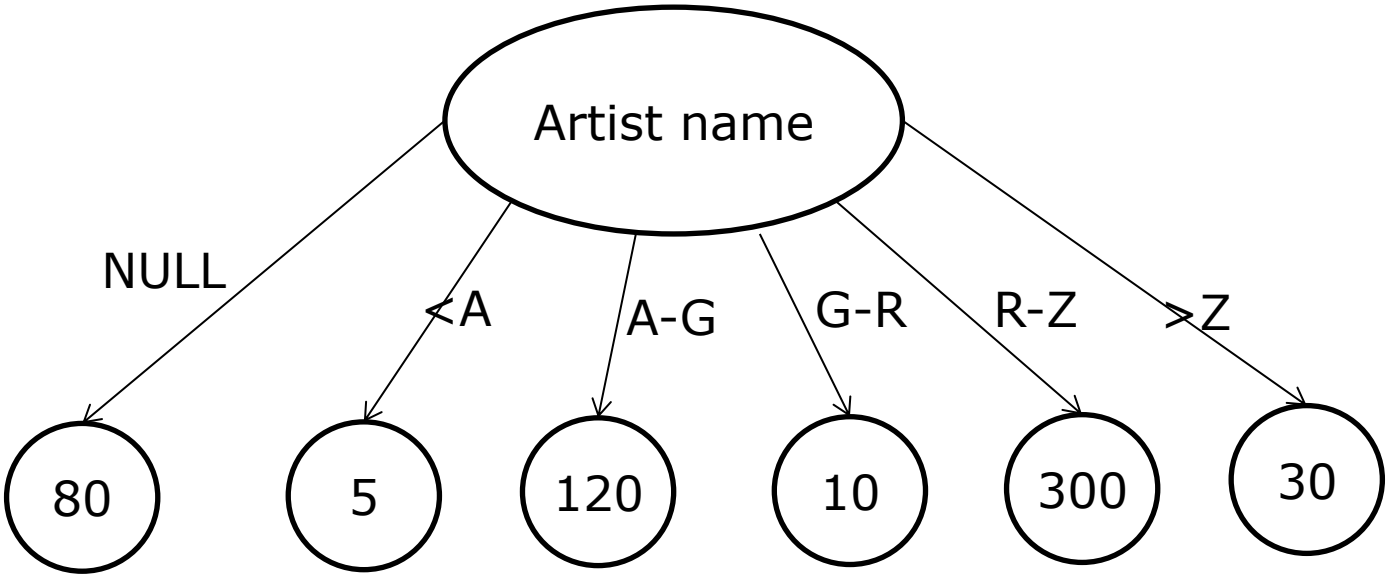
Erlernen optimaler Bäume

- Wähle Hashfunktion mit wenigsten getrennten Duplikaten
  - Schätze Größe aller Kindknoten
  - Wiederholung für alle großen Kindknoten



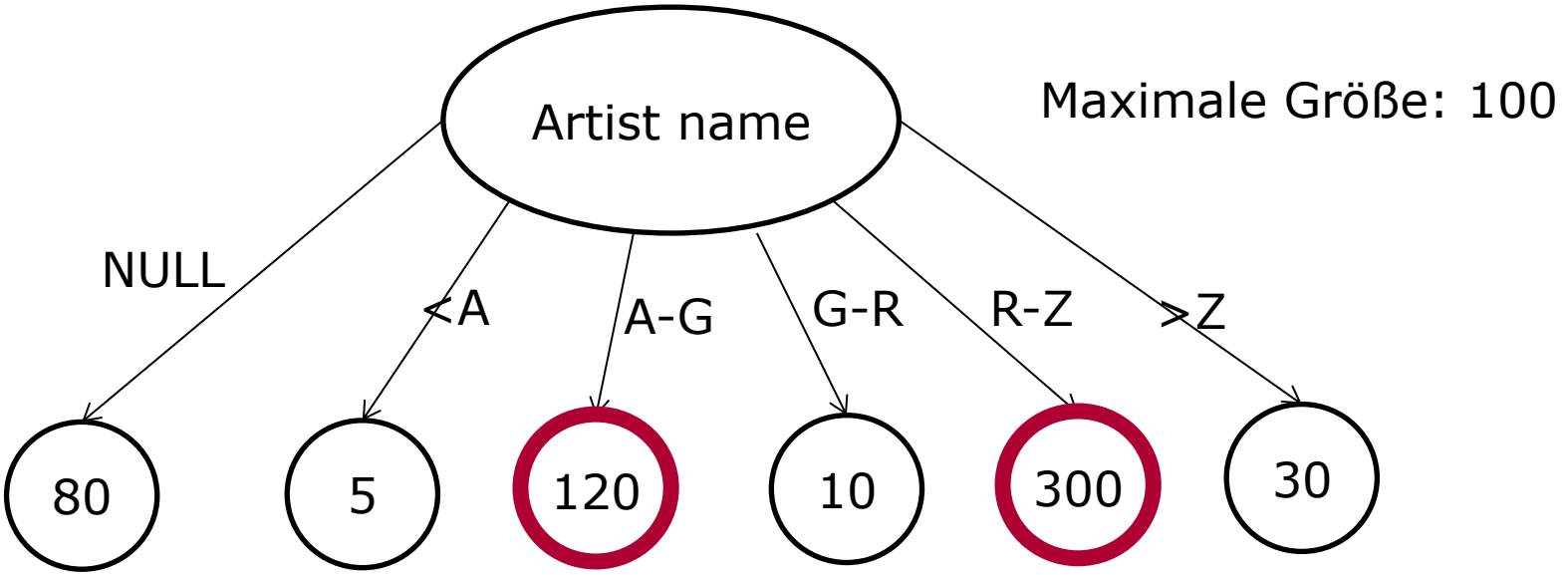
Erlernen optimaler Bäume

- Wähle Hashfunktion mit wenigsten getrennten Duplikaten
- Schätze Größe aller Kindknoten
- Wiederholung für alle großen Kindknoten



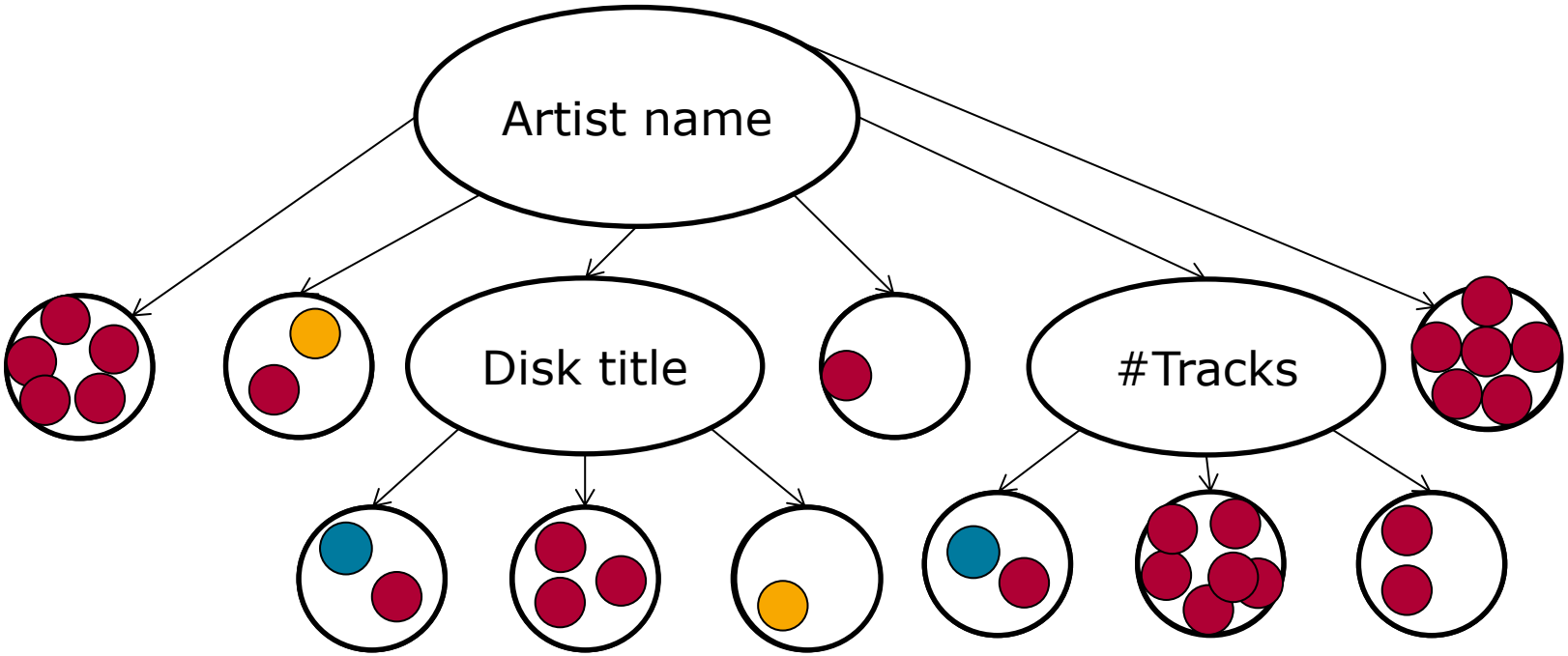
Erlernen optimaler Bäume

- Wähle Hashfunktion mit wenigsten getrennten Duplikaten
- Schätze Größe aller Kindknoten
- Wiederholung für alle großen Kindknoten



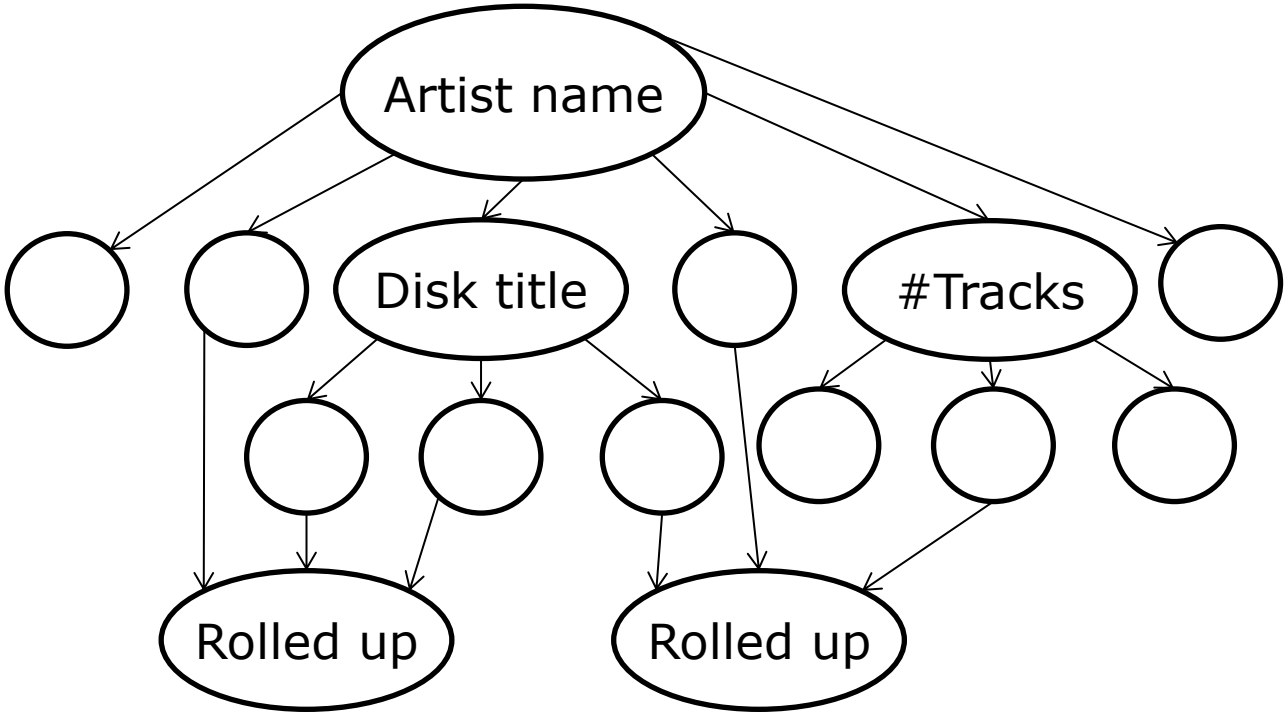
Rollup

- Problem: es können viele kleine Blöcke entstehen
  - Unnötig viele Duplikate getrennt
  - Zusammenführen von Knoten kann Duplikate wieder zusammenbringen



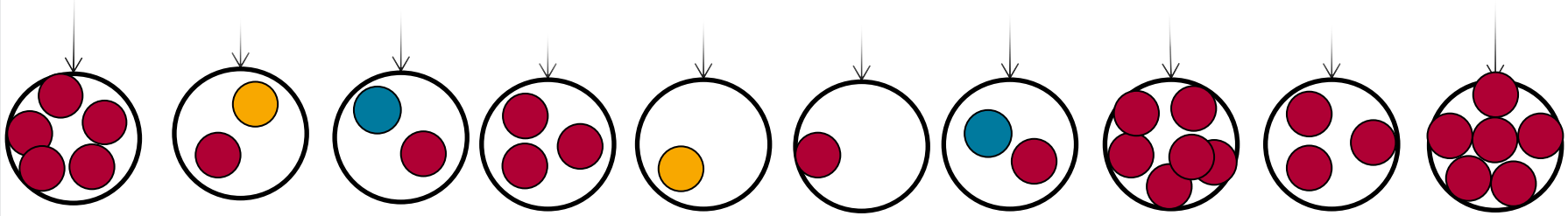
Rollup

- Voraussetzung
  - Kompletter Baum vorhanden
  - Geschätzte Größe aller Blattknoten bekannt



Rollup

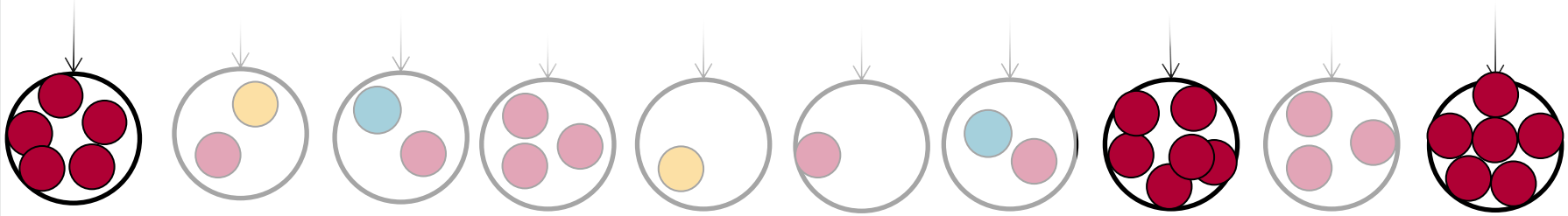
- Große Blöcke ignorieren
- Suche Blöcke zum Zusammenführen



Maximale Größe: 5

Rollup

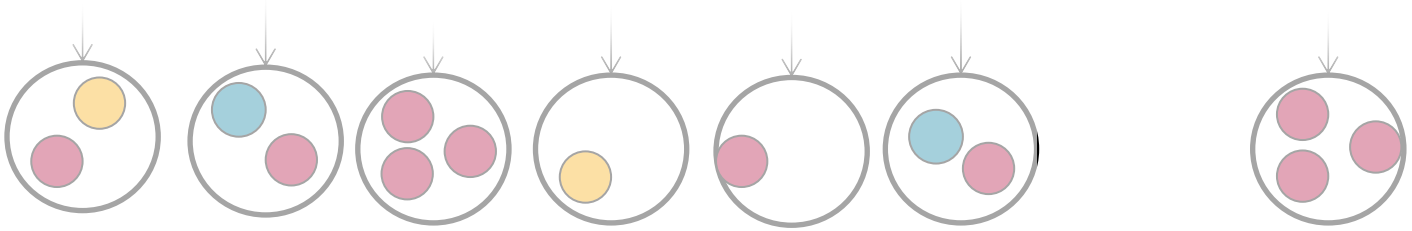
- Große Blöcke ignorieren
  - Suche Blöcke zum Zusammenführen



Maximale Größe: 5

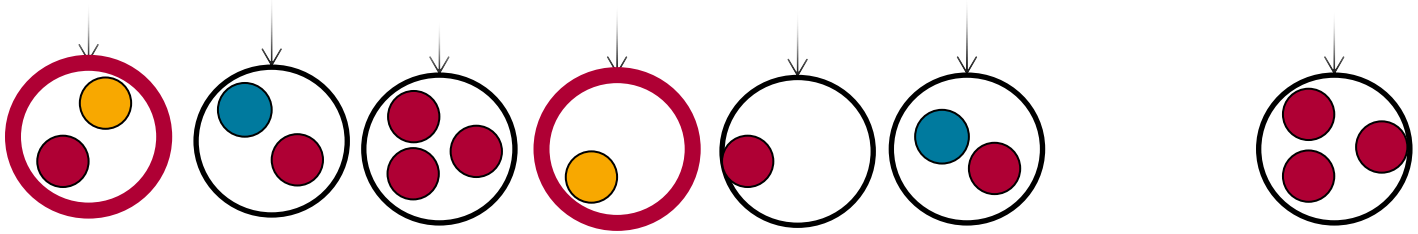
Rollup

- Große Blöcke ignorieren
- Suche Blöcke zum Zusammenführen



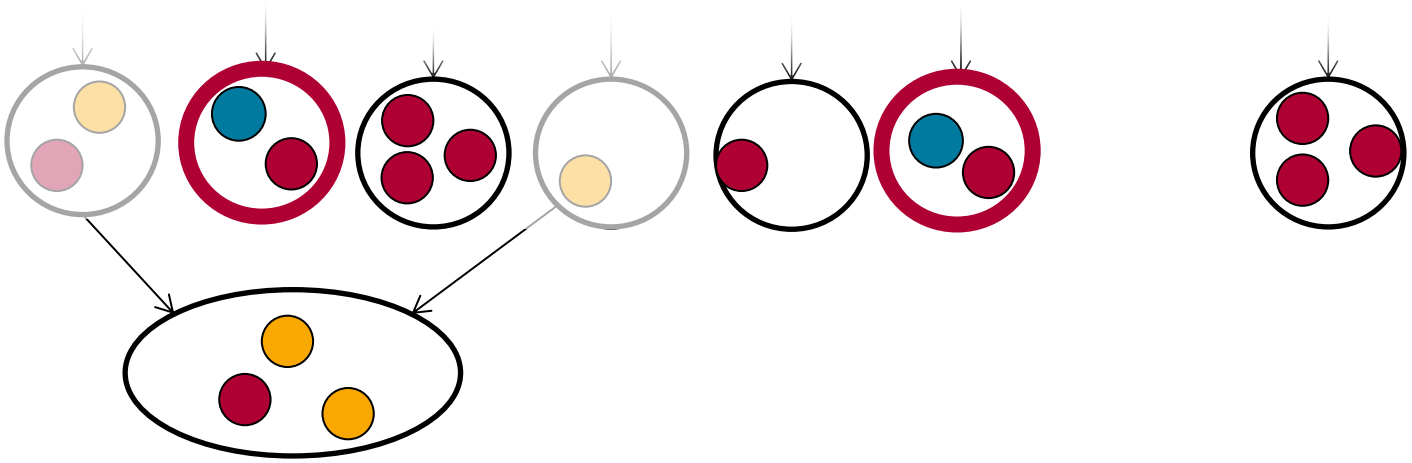
Rollup

- Große Blöcke ignorieren
- Suche Blöcke zum Zusammenführen



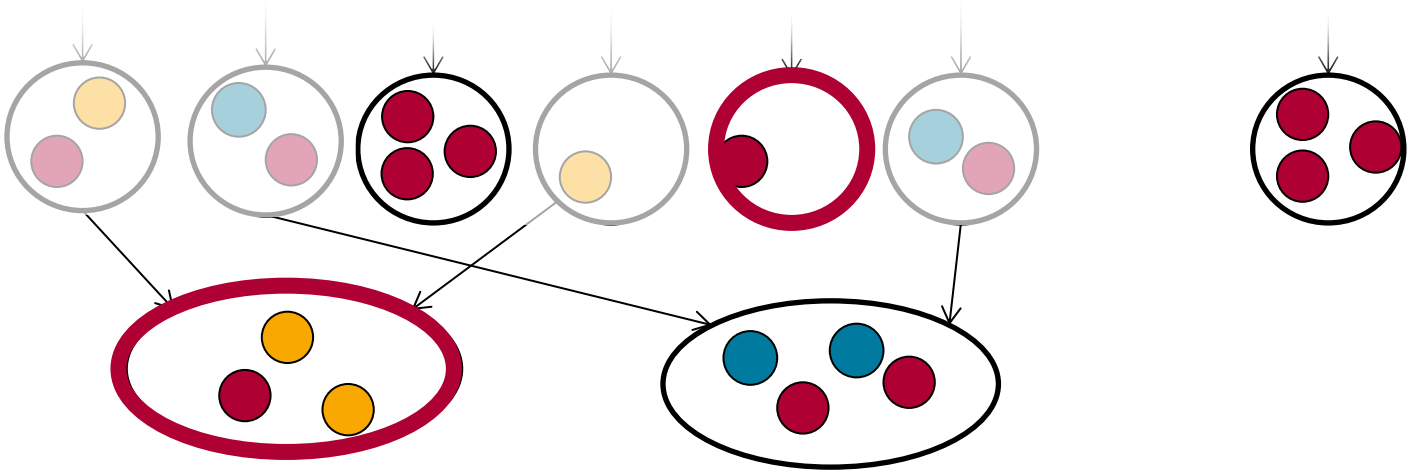
Rollup

- Große Blöcke ignorieren
- Suche Blöcke zum Zusammenführen



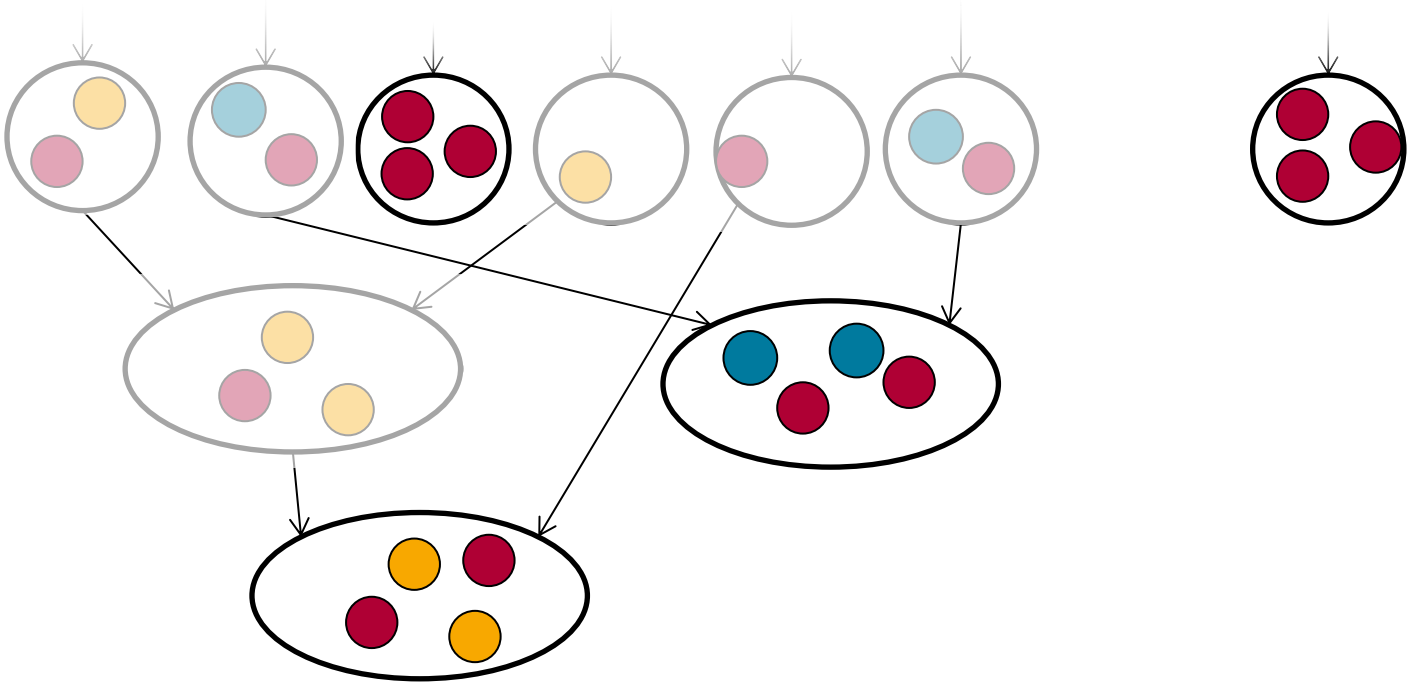
Rollup

- Große Blöcke ignorieren
- Suche Blöcke zum Zusammenführen



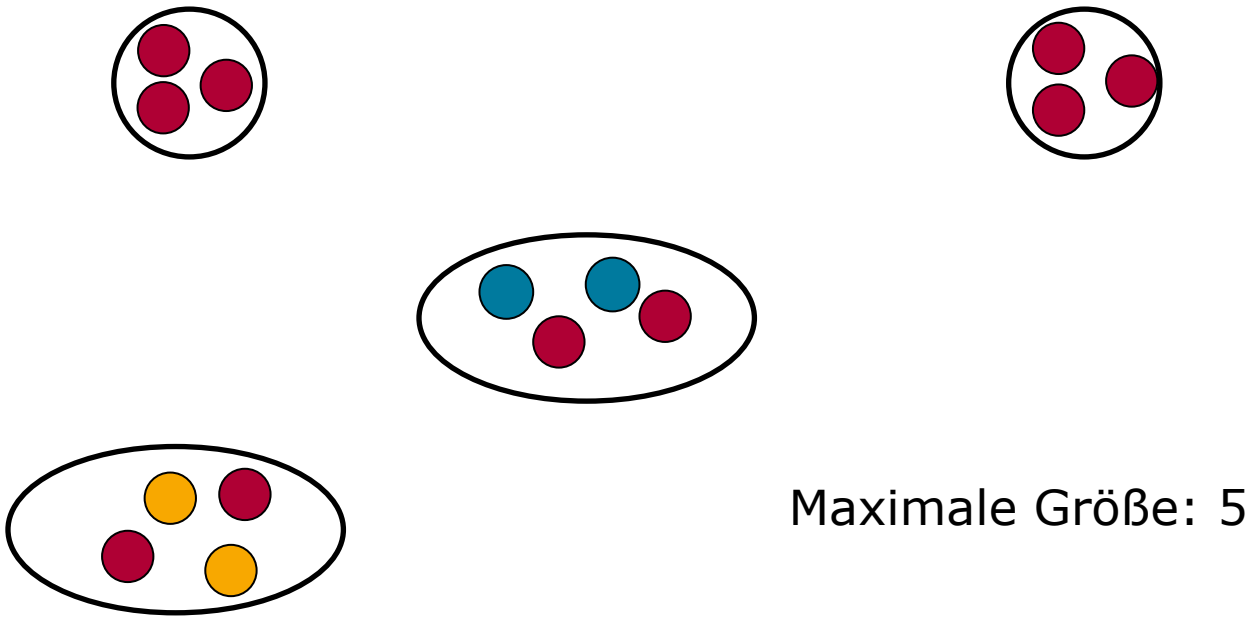
Rollup

- Große Blöcke ignorieren
- Suche Blöcke zum Zusammenführen



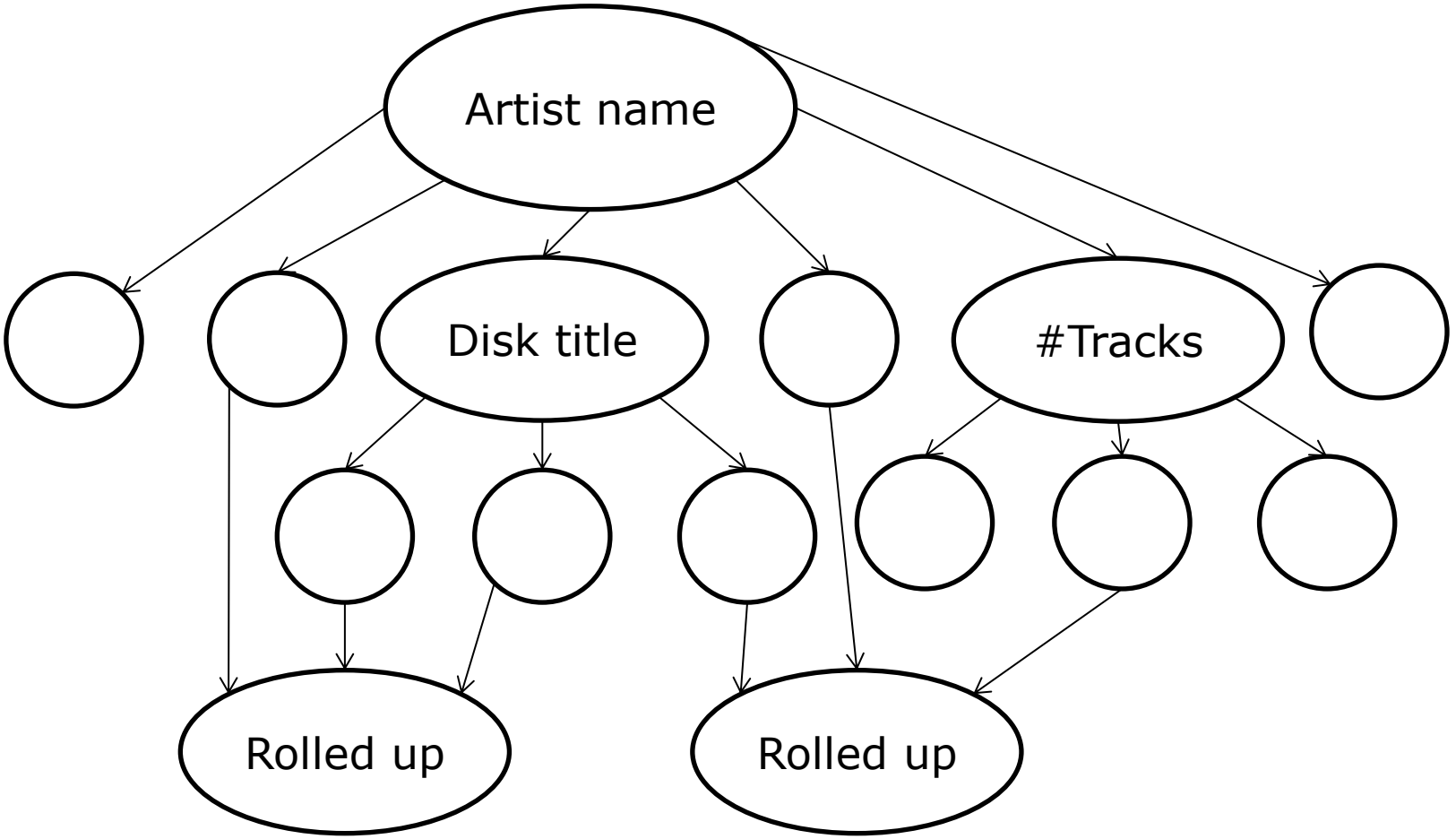
Rollup

- Große Blöcke ignorieren
- Suche Blöcke zum Zusammenführen



Maximale Größe: 5

Rollup - Ergebnis

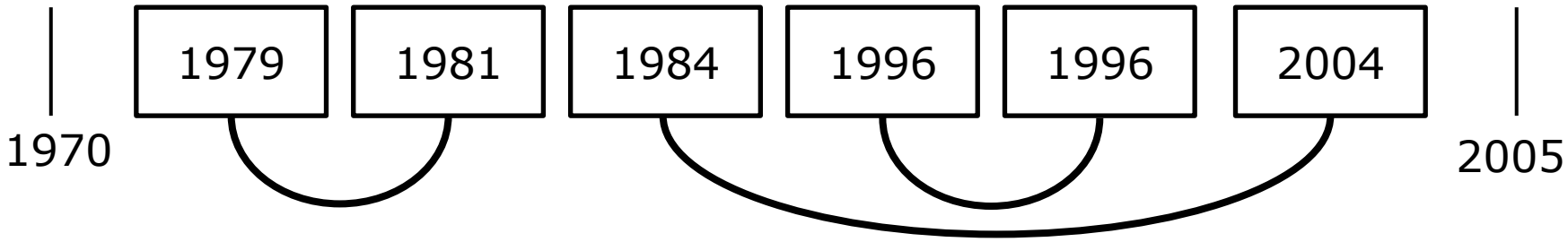


Drill-Down

- Woher kommen die Hashfunktionen?
  - Manuell überlegen
  - Automatisch generieren
- Drill-Down Algorithmus generiert Hashfunktion basierend auf einem Attribut der Eingabedaten

Drill-Down

- Attribut in Intervalle aufteilen
  - Unterliegt Ordnungsrelation
  - Wertebereich bekannt
    - z.B. Jahreszahlen zwischen 1970 und 2005

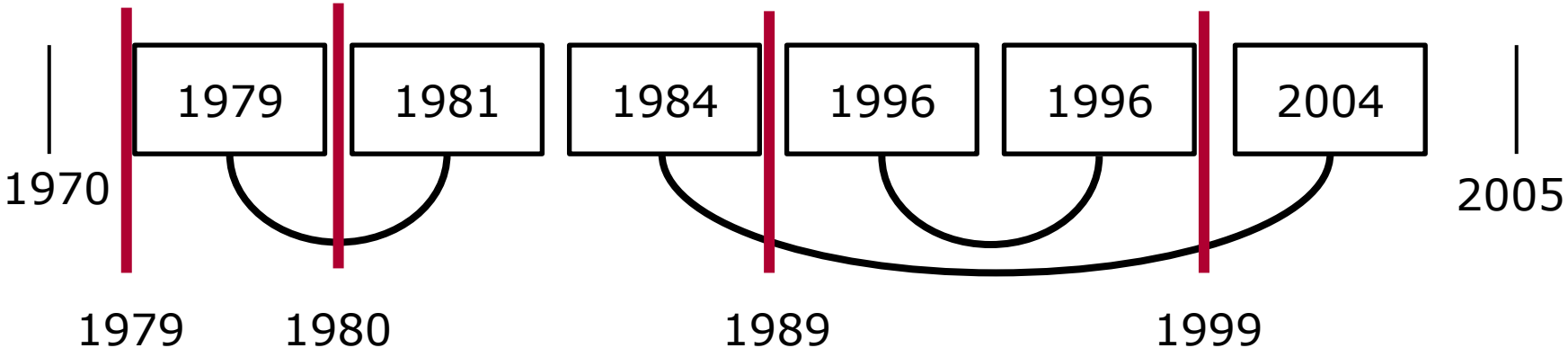


Kostengrenze: 10 Jahre pro Intervall

Drill-Down

- Möglichst keine Duplikate trennen
- Intervalle enden an einem Duplikat oder an Kostengrenze

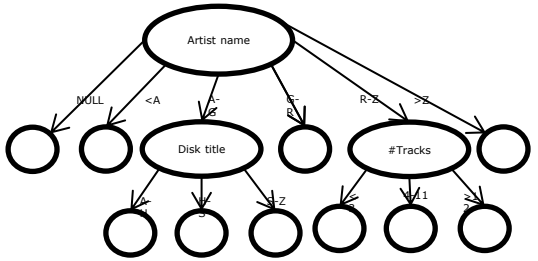
- Errechne optimale Unterteilung durch rekursiven Ansatz



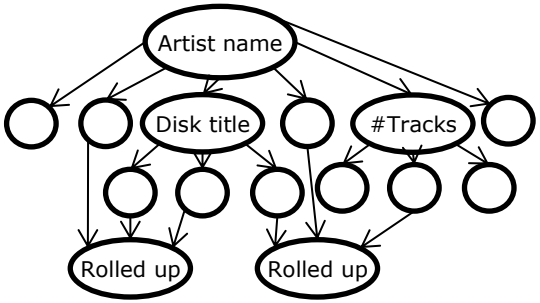
Kostengrenze: 10 Jahre pro Intervall

Zusammenfassung

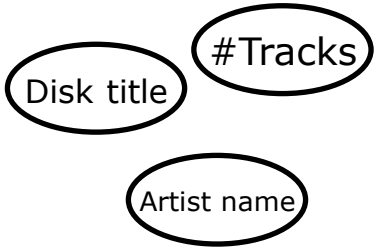
- Blocking-Algorithmus zur Nutzung in parallelen Umgebungen
  - Repräsentation der Hash-Funktion als Baumstruktur



  - Vermeidung kleiner Blöcke durch Rollup



  - Automatisches Generieren von Hash-Funktionen



Offene Probleme und Erweiterungen

- Erfolg hängt von Qualität der Trainingsdaten ab
  - Repräsentative Verteilung und Größe
  - Möglichst alle Duplikatarten vorhanden, wie z.B.
    - Schreibfehler
    - komplett unterschiedliche Felder

Watazumi Doso; Music Of Japan ; 21

Watazumi Doso; The Art Of Japanese Bamboo Flute ; 21

Offene Probleme und Erweiterungen

- Erlernen optimaler Bäume: Größenschätzung der Knoten nicht trivial
- Drill-Down: Ordnungsrelation nicht immer trivial
- Hohe Komplexität durch viele konfigurierbare Parameter
  - Maximale Knotengröße für Erstellung des Baumes
  - Maximale Knotengröße für Rollup
  - Kostengrenze für Drill Down
  - Wahl der Hashfunktionen (wenn ohne Drill Down)

- Statt komplexer Optimierung des Baumes wäre auch Kombination mit „Multi-pass Sorted Neighborhood Blocking“-Paper denkbar

Referenzen

- Sarma, Anish Das, et al. "CBLOCK: An Automatic Blocking Mechanism for Large-Scale De-duplication Tasks." *arXiv preprint arXiv:1111.3689* (2011).
- Beispiele von www.freedb.org

Übersichtsfolie zu Paper

- Detaillierte Algorithmenbeschreibungen
  - Baumgenerierung: Kapitel 4.4 „Greedy Algorithm“, Seite 7f.
  - Rollup: Kapitel 5 „Rolling up small canopies“, Seite 8f.
  - Drill-Down: Kapitel 6 „Drill-Down Problem“, Seite 9ff.
- Spezialfälle des Baumes
  - Vorstellung: Kapitel 4.2.1 „Restricted languages“, Seite 7
  - Vergleich der Baumarten: Kapitel 8, Seite 12 ff.

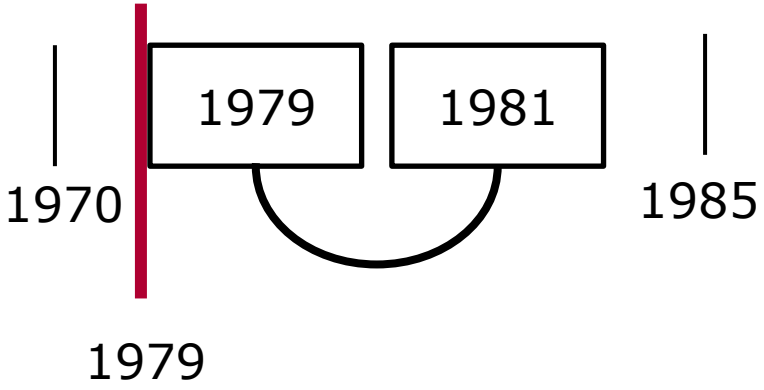
Umfang der Trainingsdaten

- Beispiel im Paper
  - Gesamtdatensatz: 140 000 Tupel
  - Trainingsdaten: 1054 Paare, also 2108 Tupel (ca. 1,5%)
- Unser Datensatz
  - Gesamtdatensatz: 1 900 000 Tupel
  - Geeignete Trainingsdatengröße: ca. 14 000 Paare

Ausführliches Drill-Down Beispiel

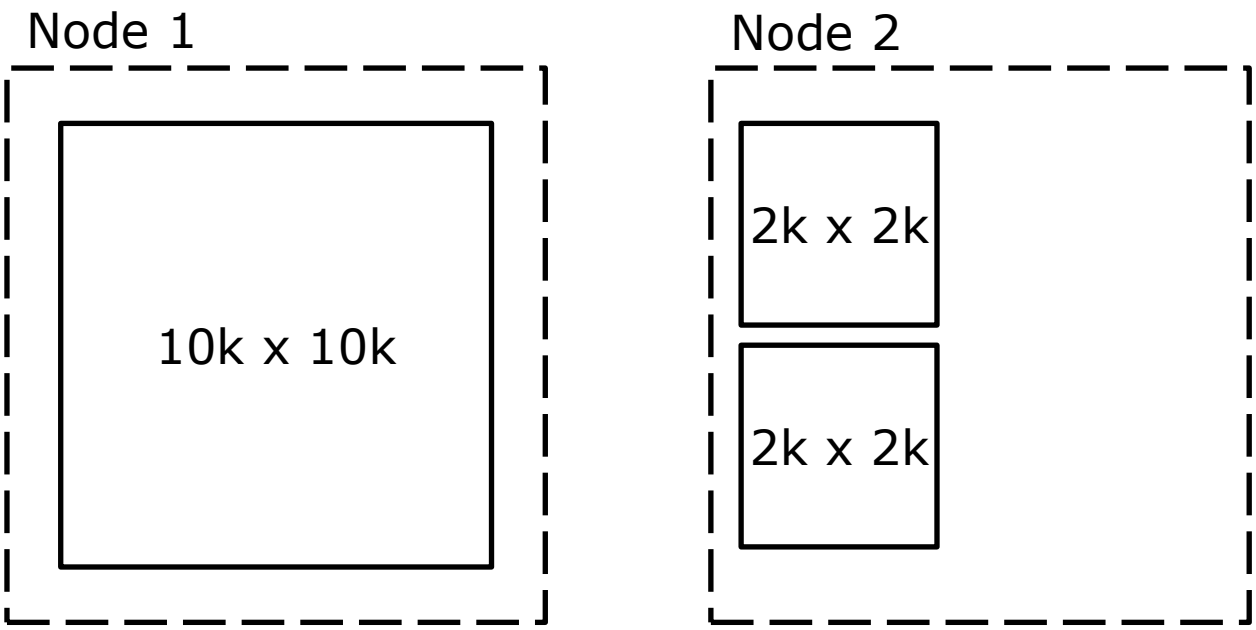
- Möglichst keine Duplikate auseinander reißen
- Blöcke enden nach einem Duplikat oder an Kostengrenze

- Bsp. Mit Wertebereich 1970 – 2005, maximale Kosten 10:



Rollup - Auslastung

- Maximum der Rechenzeit durch größten Block gegeben:



Rollup - Auslastung

- Maximum der Rechenzeit durch größten Block gegeben: