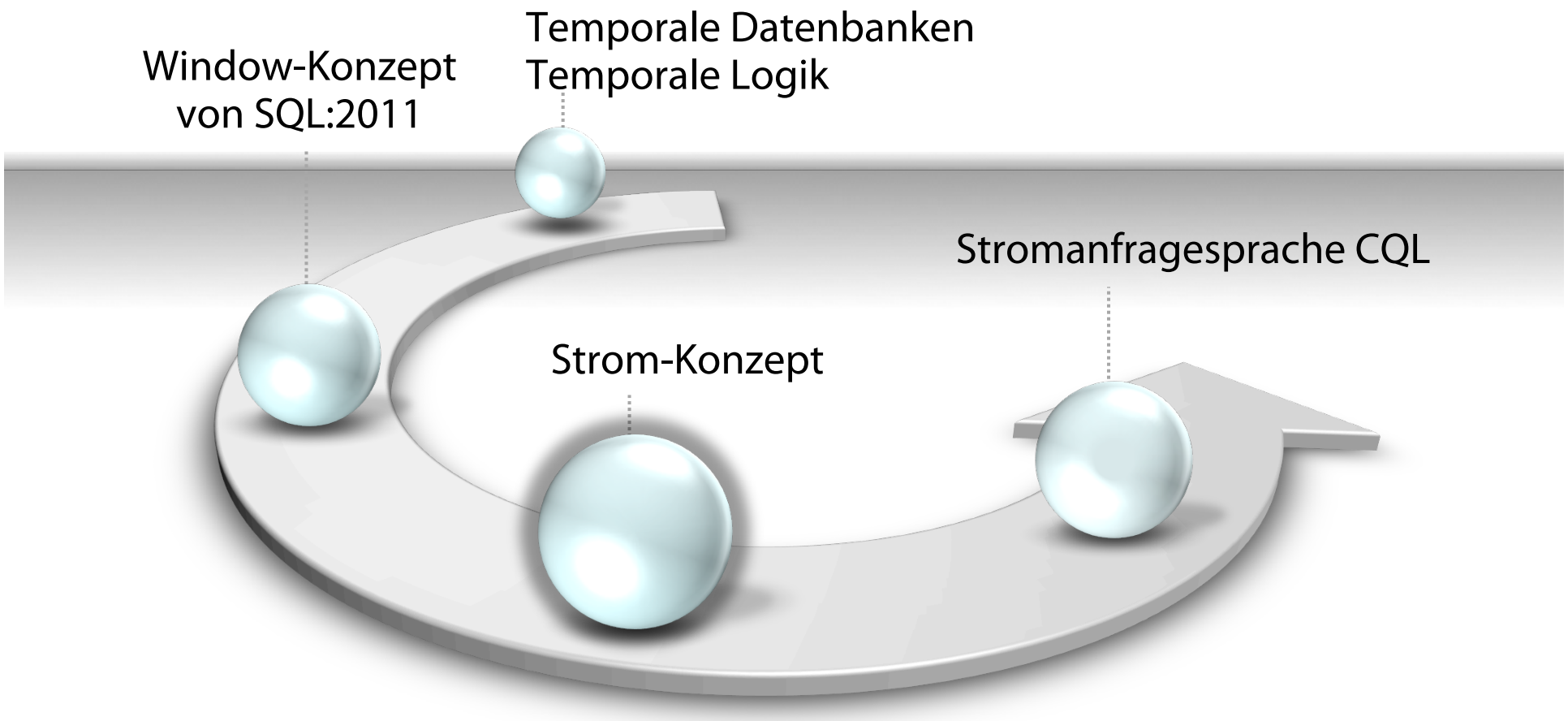
Non-Standard-Datenbanken

Stromdatenbanken

Prof. Dr. Ralf Möller
Universität zu Lübeck
Institut für Informationssysteme

Non-Standard-Datenbanken

Von temporalen Datenbanken zu Stromdatenbanken



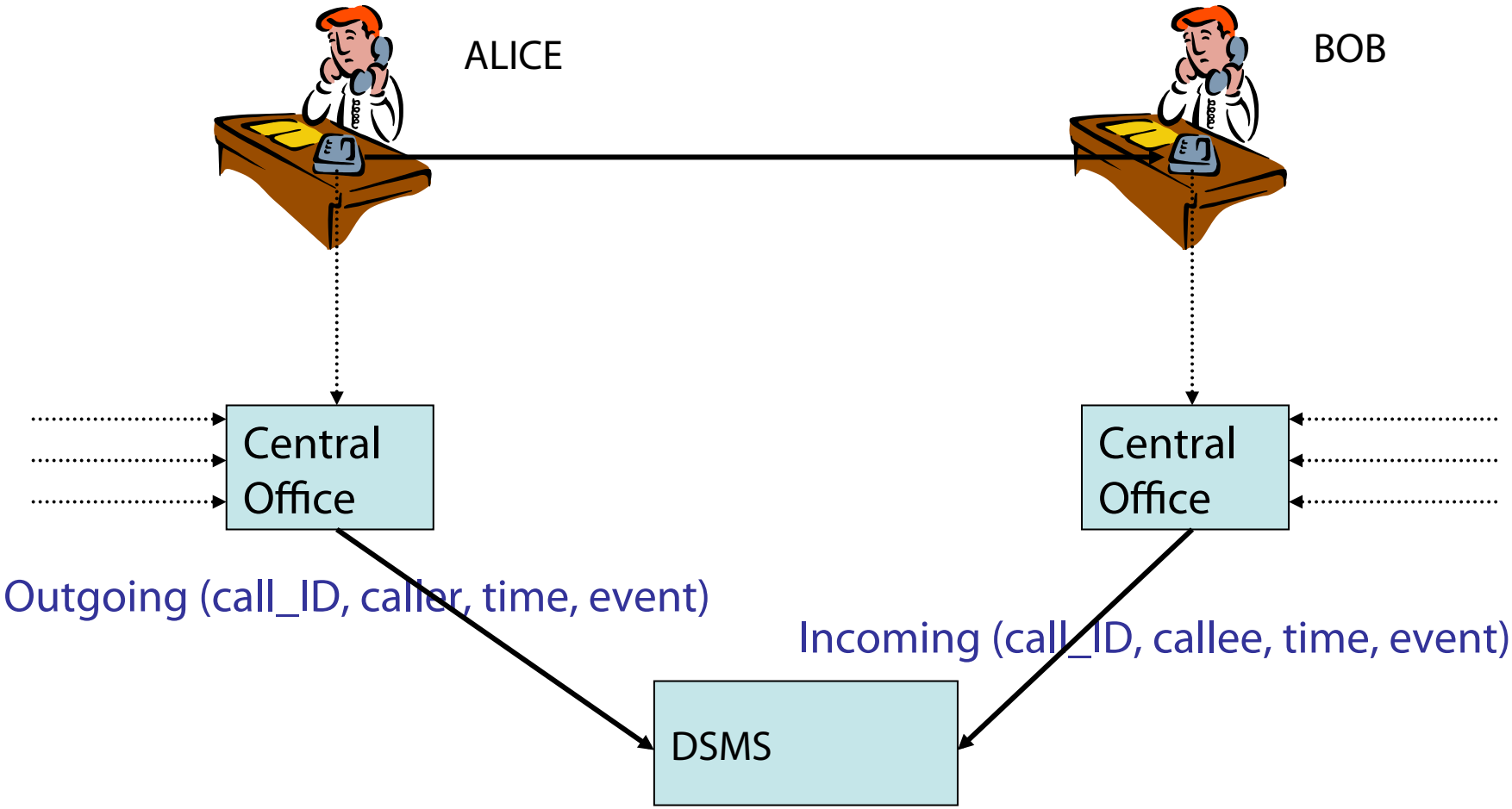
Datenströme: Motivation

- Traditionelle DBMS – Daten in endlichen persistenten Dateneinheiten gespeichert (z.B. in Tabellen, XML-Graphen, …) ggf. mit Anwendungszeit- bzw. Systemzeitattributen
- Neue Anwendungen – Daten als kontinuierliche, geordnete Ströme von Tupeln aufgefasst
  - IP-Netzwerkverbindungen
  - Telefonverbindungen
  - Finanzielle Transaktionen
  - Sensornetzwerkorganisation (z.B. in der Produktion, im Produkt, im Krankenhaus, …)
  - Weblogs und Klickströme (clickstreams)
  - Internet der Dinge (Internet of things)

Stromdatenbanken vs. Stromverarbeitungs-SW

- Stromdatenbank: Verarbeitung komplexer Ereignisse (CEP)
  - Deklaratives Datenmanagement (CEP+DB-Archivierung)
  - Deklarative Anfragesprache
  - Projekte: Aurora/Borealis (Brandeis, Brown, MIT), Odysseus (Oldenburg), PIPES (Marburg), STREAM (Stanford), TelegraphCQ (Berkeley),
  - Komm. Anbieter: Microsoft, Oracle, SAP (Sybase Event Stream Processor), Software-AG (ex Apama), TIBCO Software (ex Streambase, eine kommerzielle Ausprägung von Aurora), ...
- Softwarearchitekturen zur Stromverarbeitung
  - Manuelles Aufsetzen von Datenflussnetzen
  - Prozedurale Definition von Algorithmen und Datenstrukturen
  - Open-Source Software: Apache Storm, Apache SPARK ...
  - Komm. Anbieter: Gigaspaces, ...

Beispiel 1: Telefondatenauswertung



event = **START** or **END**

B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom. Models and Issues in Data Stream Systems, Invited paper in Proc. of PODS 2002, June **2002**

Anfrage 1 (SELF-JOIN)

- Find all outgoing calls longer than 2 minutes

```
SELECT   O1.call_ID, O1.caller
FROM     Outgoing O1, Outgoing O2
WHERE    (O2.time – O1.time > 2
         AND  O1.call_ID = O2.call_ID
         AND  O1.event = START
             AND  O2.event = END)
```

- Ergebnis wächst ohne Begrenzung
- Ergebnis als Datenstrom bereitstellbar
- Frühestmögliche Ergebnisbereitstellung:
  Für Einzelverbindung steht Ergebnis nach 2 min fest,
  auch ohne END

Anfrage 2 (JOIN)

- Pair up callers and callees

```
SELECT   O.caller, I.callee
FROM     Outgoing O, Incoming I
WHERE    O.call_ID = I.call_ID
```

- Ergebnis kann als Datenstrom bereitgestellt werden
- Unbegrenzter temporärer Speicher notwendig …
- ... wenn Ströme nicht quasi-synchronisiert sind

Anfrage 3 (Gruppierung und Aggregation)

- Total connection time for each caller

```
SELECT      O1.caller, sum(O2.time – O1.time)
FROM        Outgoing O1, Outgoing O2
WHERE       (O1.call_ID = O2.call_ID
            AND O1.event = START
                AND  O2.event = END)
GROUP BY    O1.caller
```

- Ergebnis kann nicht als Strom (ohne Überschreibung) dargestellt werden
  - Ausgabeaktualisierung?
  - Aktueller Wert auf Anforderung?
  - Speicherverbrauch?

Beispielanwendung 2

- Verkehrsüberwachung
  - Datenformat

```
HighwayStream( lane, speed, length, timestamp )
```

  - Zeitstempel explizit
  - Kontinuierlicher Datenfluss
    ➔ Strom von Daten
    - Variable Datenraten
    - Zeit- und Ortsabhängig
  - Anfragen
    - Kontinuierlich, langlaufend

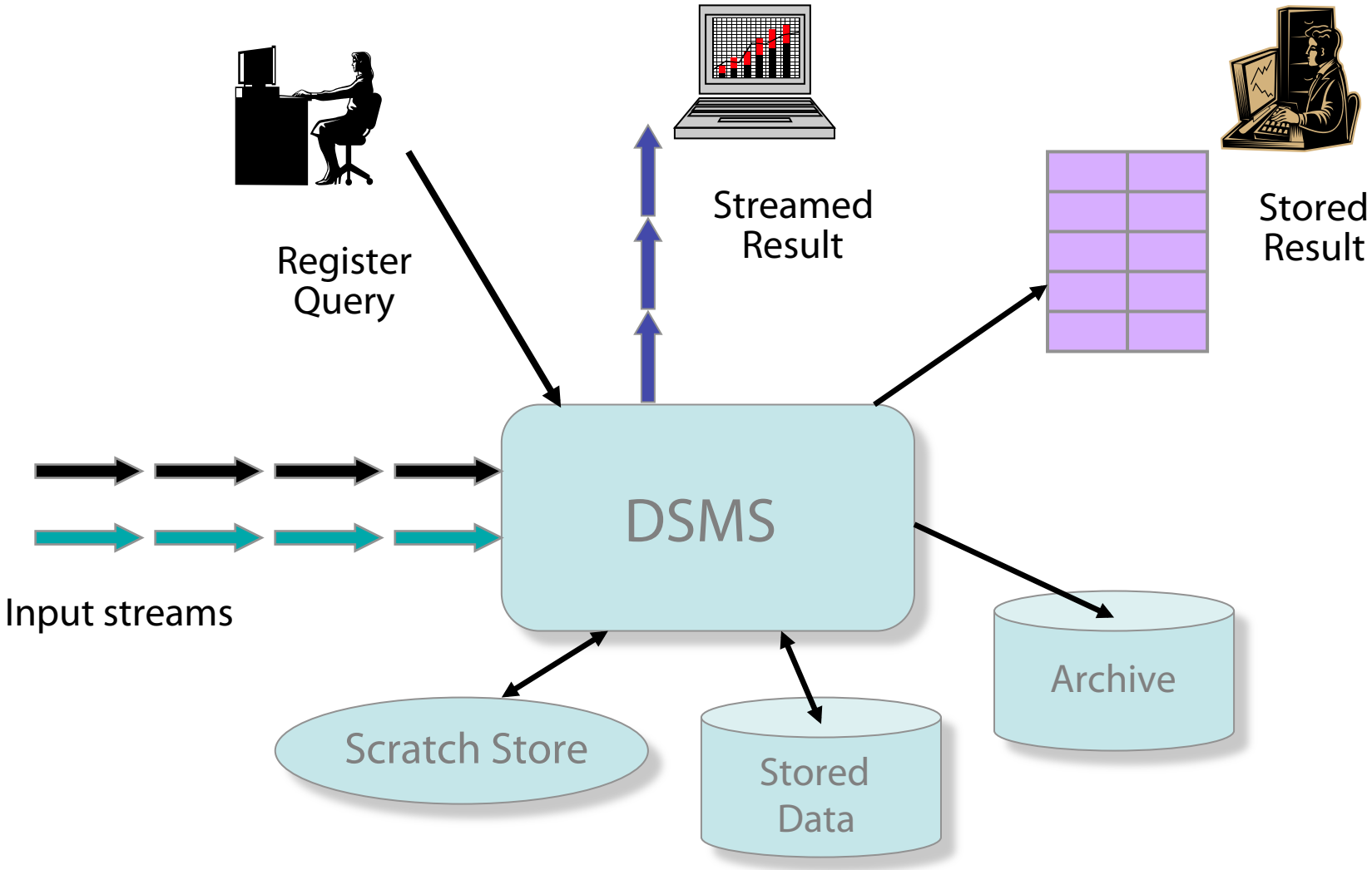
"At which measuring stations of the highway has the average speed of vehicles been below 15 m/s over the last 15 minutes ?"

siehe auch:
S. Babu, L. Subramanian, and J. Widom. A Data Stream Management System for Network Traffic Management In Proc. of NRDM 2001, May 2001

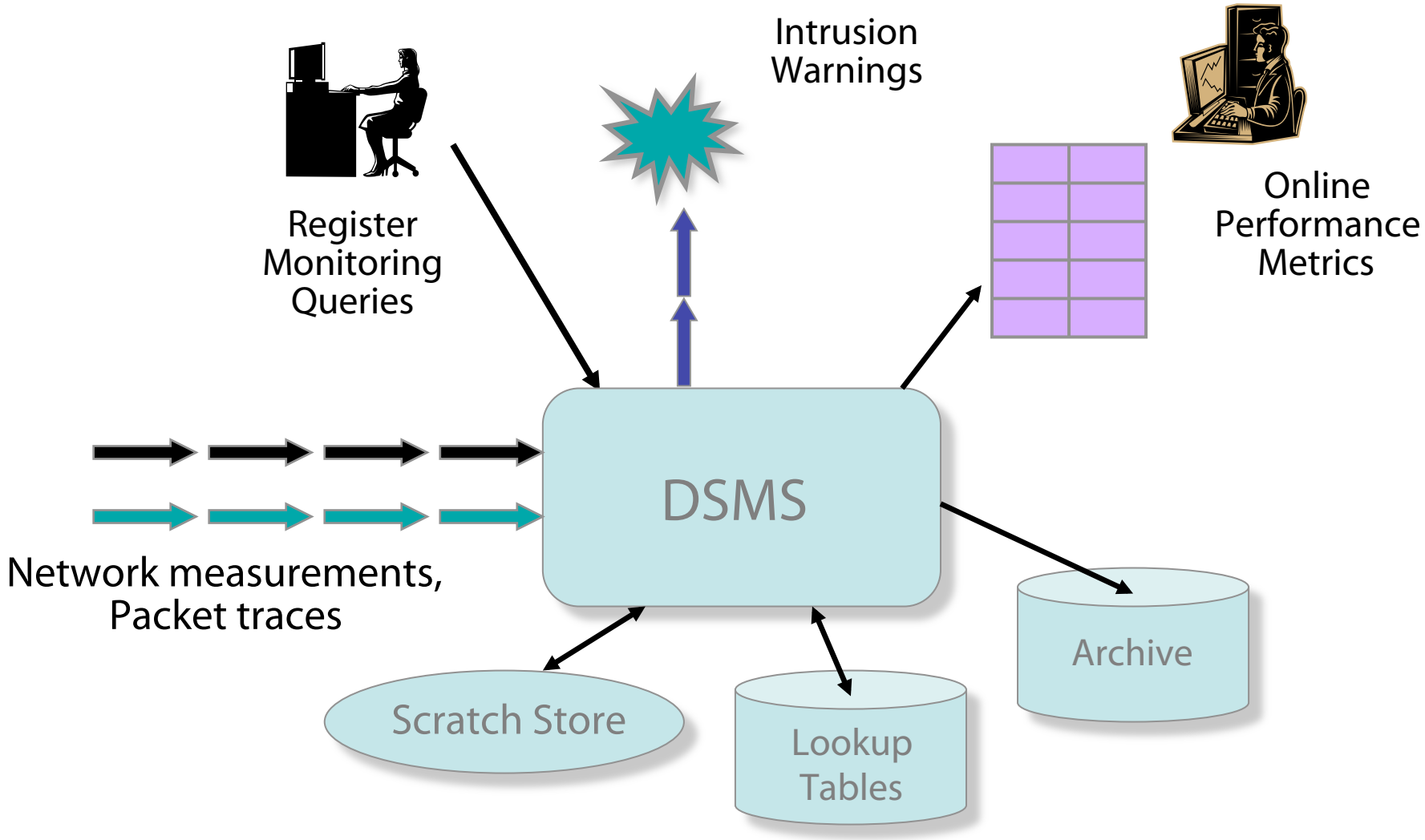
Anforderungen

- Deklarative Anfragesprache
- Ausdrucksstark wie (temporales) SQL
  - Verbund von Datenströmen bezogen auf die Zeit
  - Kombination von Datenströmen mit persistenten Datenbasen
  - Anfrageergebnisse als neue Datenströme nutzbar
- Publish/Subscribe Paradigma
  - Subscribe: Nutzer registrieren Anfragen
  - Publish: Inkrementelle Versendung von Ergebnissen
- Quality of Service (QoS)
  - Z.B. mindestens ein Ergebnistupel pro Sekunde
- Skalierbarkeit
  - Anzahl der Datenquellen
  - Höhe der Datenraten
  - Anzahl der registrierten Anfragen

Das allgemeine Anwendungsszenario

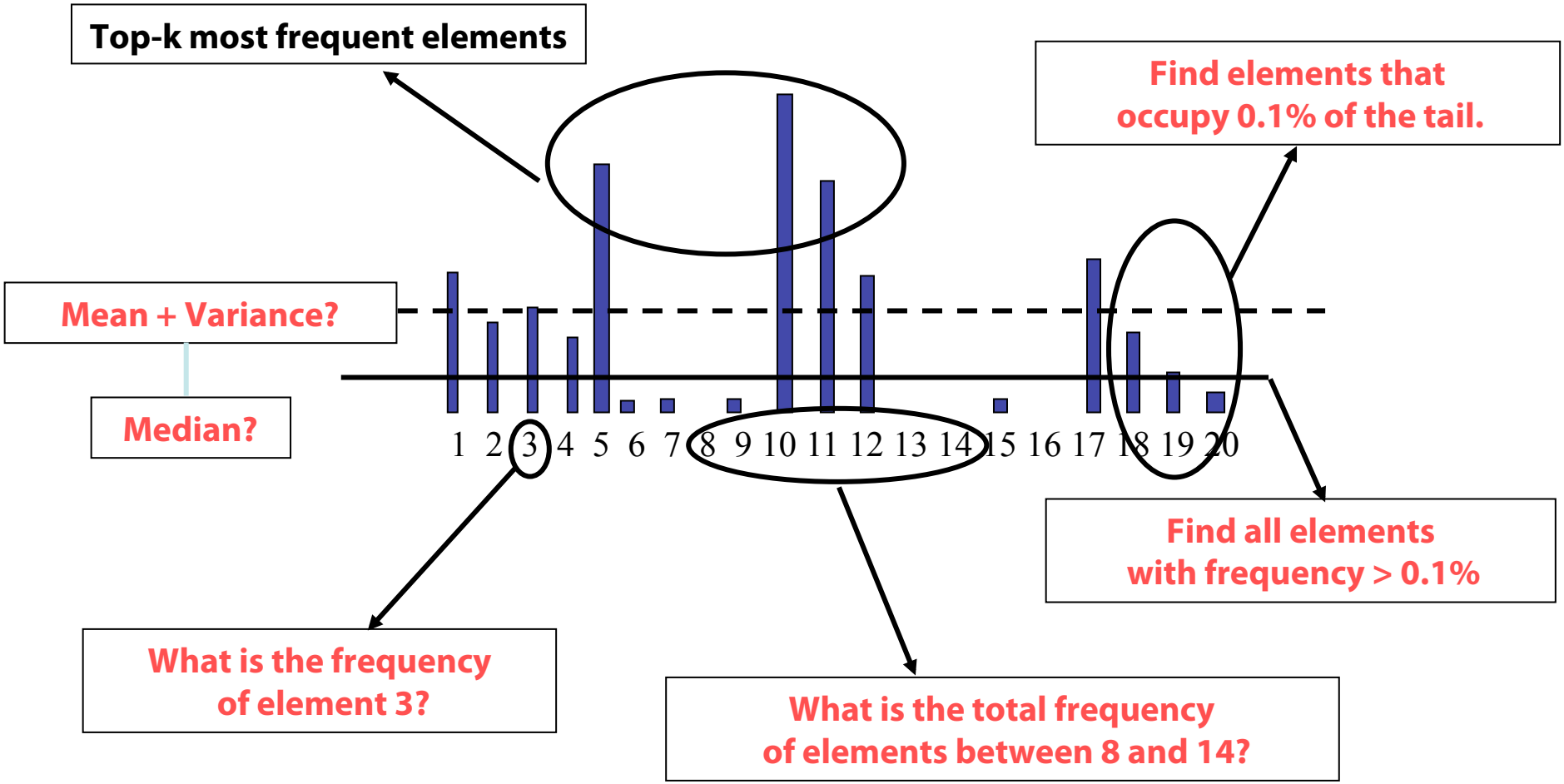


Spezielle Ausprägung: Netzwerküberwachung



Beispiel 3: Auswertung von Netzwerkdaten

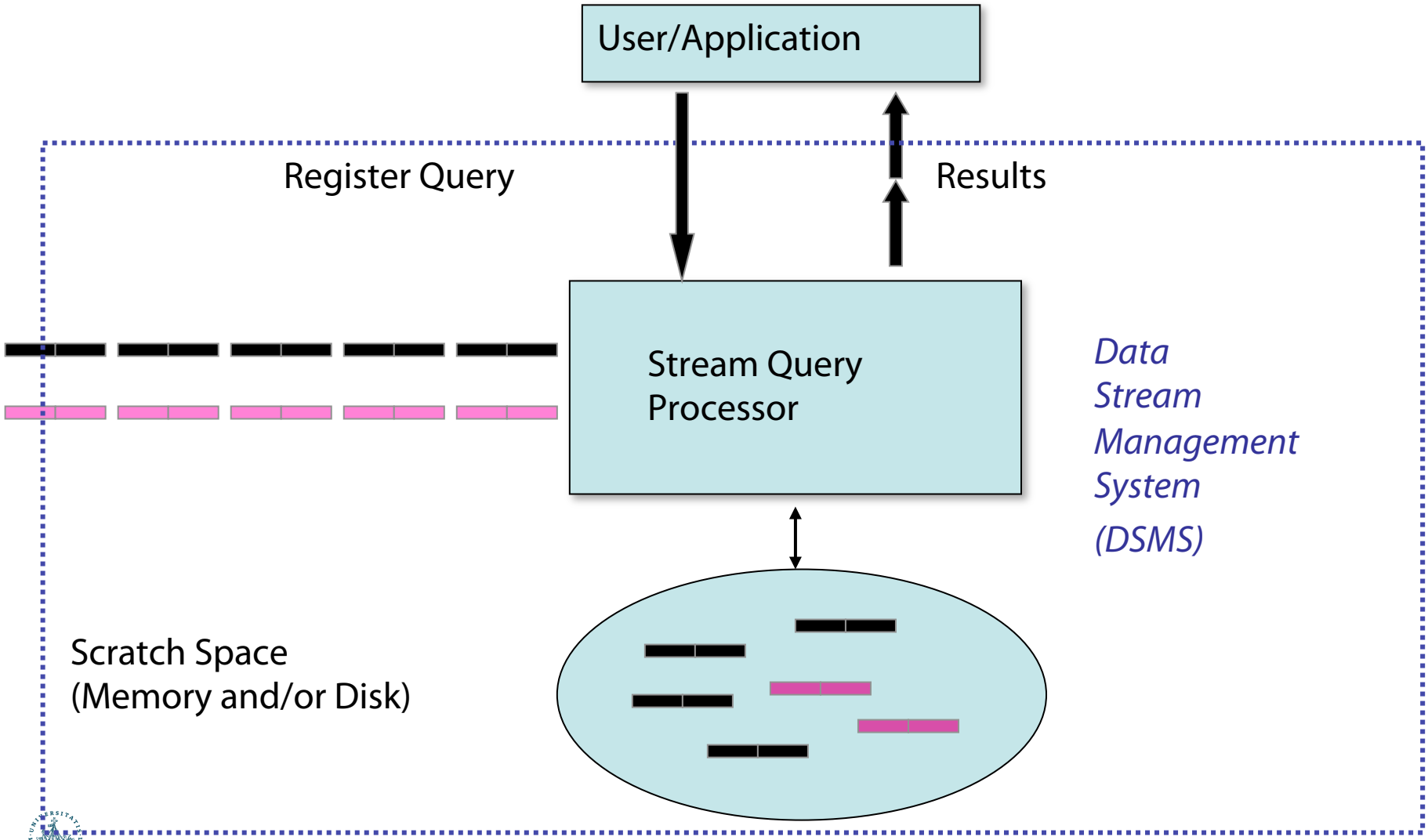
Analytics on Packet Headers – IP Addresses



Rajeev Motwani 2003

How many elements have non-zero frequency?

Data Stream Management System



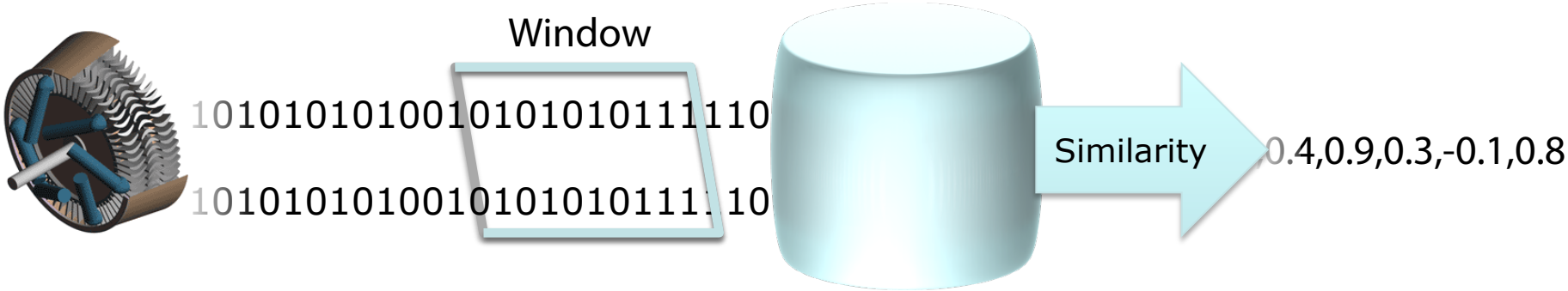
Beispiel 4: Sensordatenauswertung

- Problem

  Finde Ähnlichkeiten zwischen Messungen zweier Temperatursensoren

- Solution

  Bestimme **Pearson Korrelationskoefficient** oder **Cosinus-Dinstanz** zwischen zwei Messungen, in einem Zeitfenster



Ähnlichkeitsmaße

- Zu jedem Zeitpunkt enthalten die Vektoren $\mathbf{w_1}$, $\mathbf{w_2}$ die letzten Fensterwerte von stream_1, stream_2
- Kosinusähnlichkeit:

$$Sim_{\cos}(\mathbf{w_1}, \mathbf{w_2}) = \frac{\mathbf{w_1} \cdot \mathbf{w_2}}{\|\mathbf{w_1}\| \cdot \|\mathbf{w_1}\|}$$

- Korrelationskoeffizient (Pearson):

$$\rho(\mathbf{w_1}, \mathbf{w_2}) = \frac{\sigma_{\mathbf{w_1},\mathbf{w_2}}}{\sigma_{\mathbf{w_1}} \sigma_{\mathbf{w_2}}}$$

|| . || Euklidische Länge

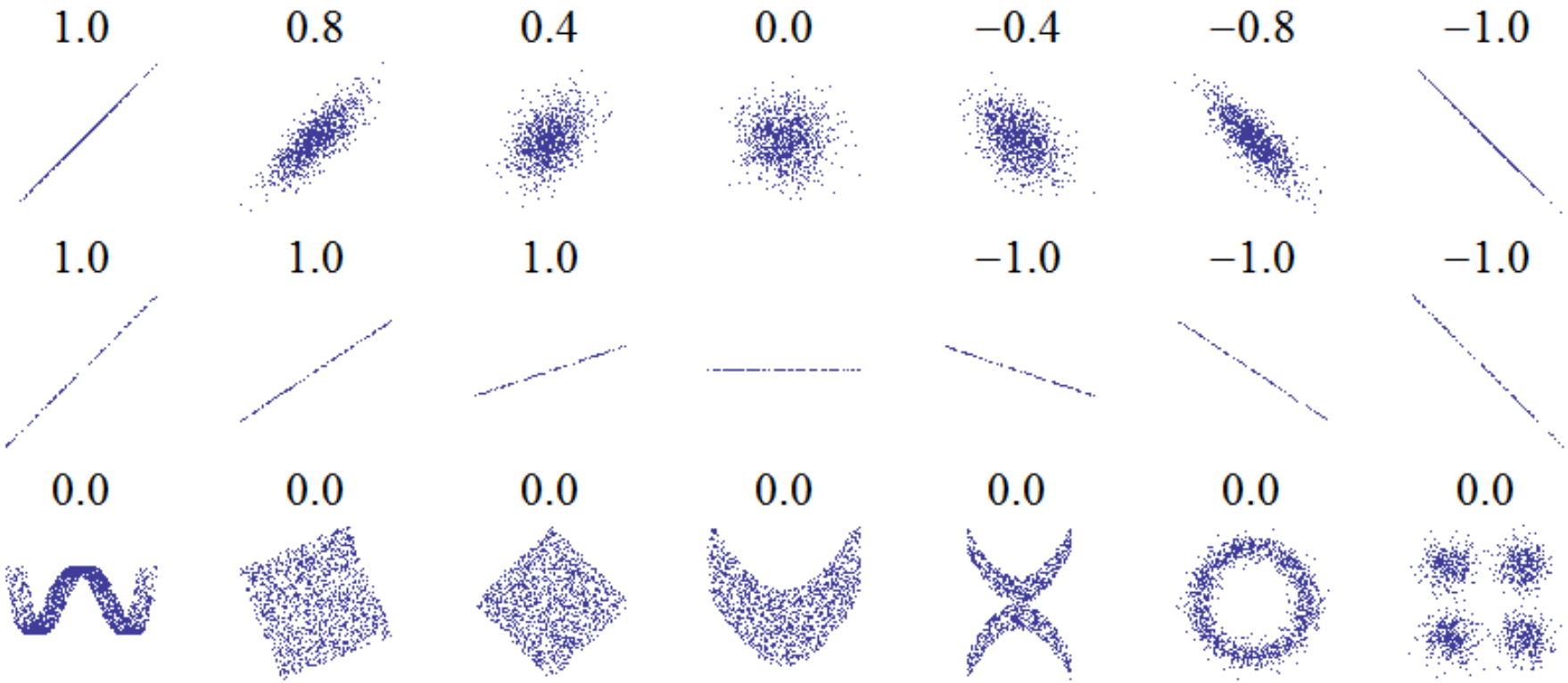
$\sigma_{\mathbf{w1,w2}}$ Kovarianz $\mathrm{Cov}(X,Y) := \mathrm{E}\left[(X - \mathrm{E}(X)) \cdot (Y - \mathrm{E}(Y))\right]$

$\sigma_{\mathbf{w1}}$ Standardabweichung $\sigma_X = \sqrt{\mathrm{Var}(X)}$ $\mathrm{Var}(X) := \mathrm{E}\left((X - \mu)^2\right)$.

$$\mu = \sum_{x \in A} x P(X = x)$$

[Wikipedia]

Korrelationskoeffizient: Visualisierung



[Wikipedia]

Fensterkonzept

Kein Einfluss von “alten” Daten auf Ergebnis

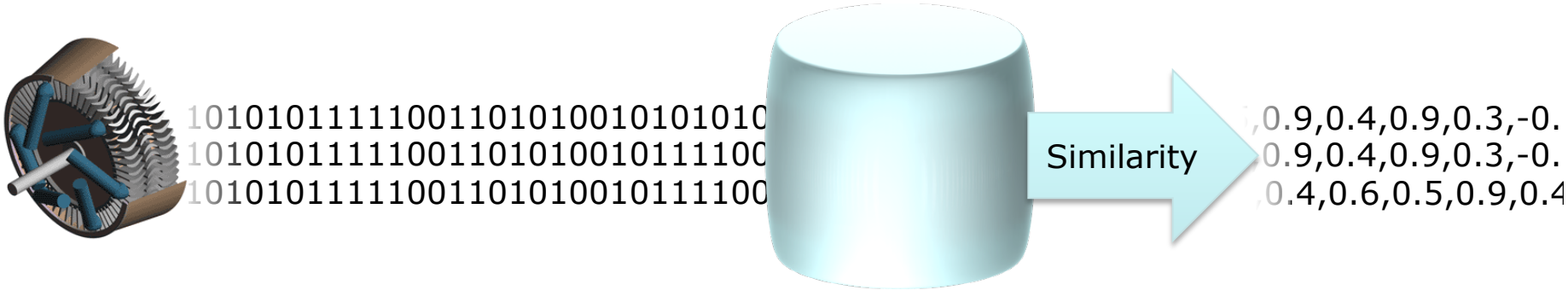
➔ Verschiebbare zeitliche Fenster

- Endliche Teilsequenzen eines unendlichen Stroms
- Anfragebeantwortung auf neueste Daten fokussiert
- Wichtig für **ausdrucksstarke Anfragen** und deren **effiziente Verarbeitung**

- Alternativen
  - Zählerbasierte Fenster (siehe auch SQL:2011)
    - FIFO-Schlange der Größe w
  - Zeitbasierte Fenster
    - t erster Zeitstempel für ein Datenelement (Zeitpunktfenster)
    - t + w Ende der Gültigkeit für ein Datenelement (Zeitintervallfenster)

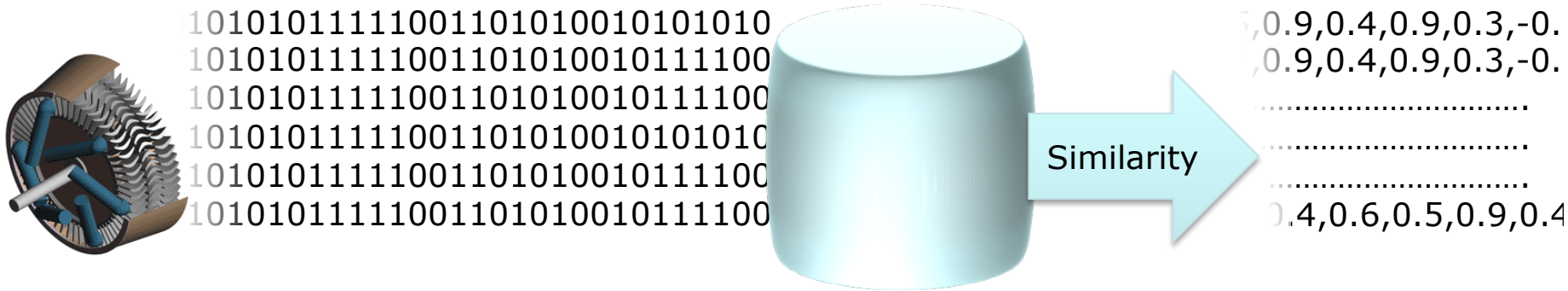
Beispiel 4: Sensordatenauswertung

- Was passiert, bei mehr als zwei Sensoren?
  - Für 3 Ströme 3 Korrelationen pro Fenster



Beispiel 4: Sensordatenauswertung

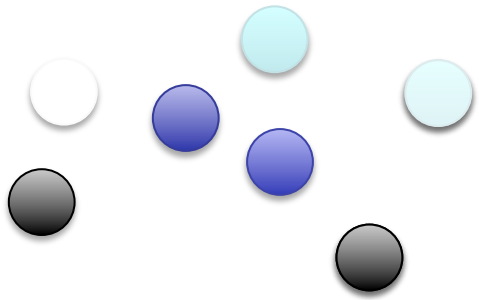
- Was passiert, bei mehr als zwei Sensoren?
  - Für 3 Ströme 3 Korrelationen pro Fenster
  - Für 6 Ströme 15 Korrelationen pro Fenster
  - Für 100 Ströme 4950 Korrelationen …
  - Quadratischer Anstieg (halbe Matrix ohne Diagonale)

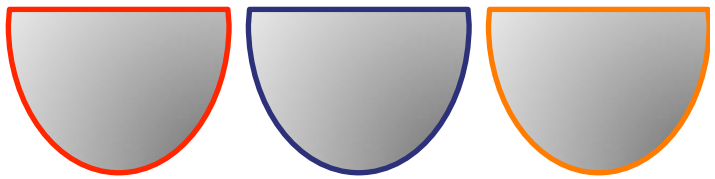


Approximation

Locality Sensitive Hashing (LSH):

- Jedes Objekt wird in Partition gehasht
- Objekte in der gleichen Partition häufig ähnlich
- Korrelation zwischen weniger Objekten berechnen
- Es gibt falsch-negative Ergebnisse (zwei ähnliche Objekten in verschiedenen Partitionen)
- Fehler kann klein gehalten werden





DBMS versus DSMS

- Persistente Relationen
- Einmalanfragen
- Wahlfreier Zugriff auf Daten
- "Unbegrenzter" Speicherplatz
- Nur aktueller Zustand zählt für Verarbeitungsergebnis
- Passive Datenverarbeitung
- Relativ langsame Aktualisierung von Daten
- Keine Realzeitanforderungen
- Korrekte Antworten erwartet
- Anfrageausführungsplan durch Anfrageprozessor auf Basis der geg. Daten und gem. des physikalischen Designs bestimmt

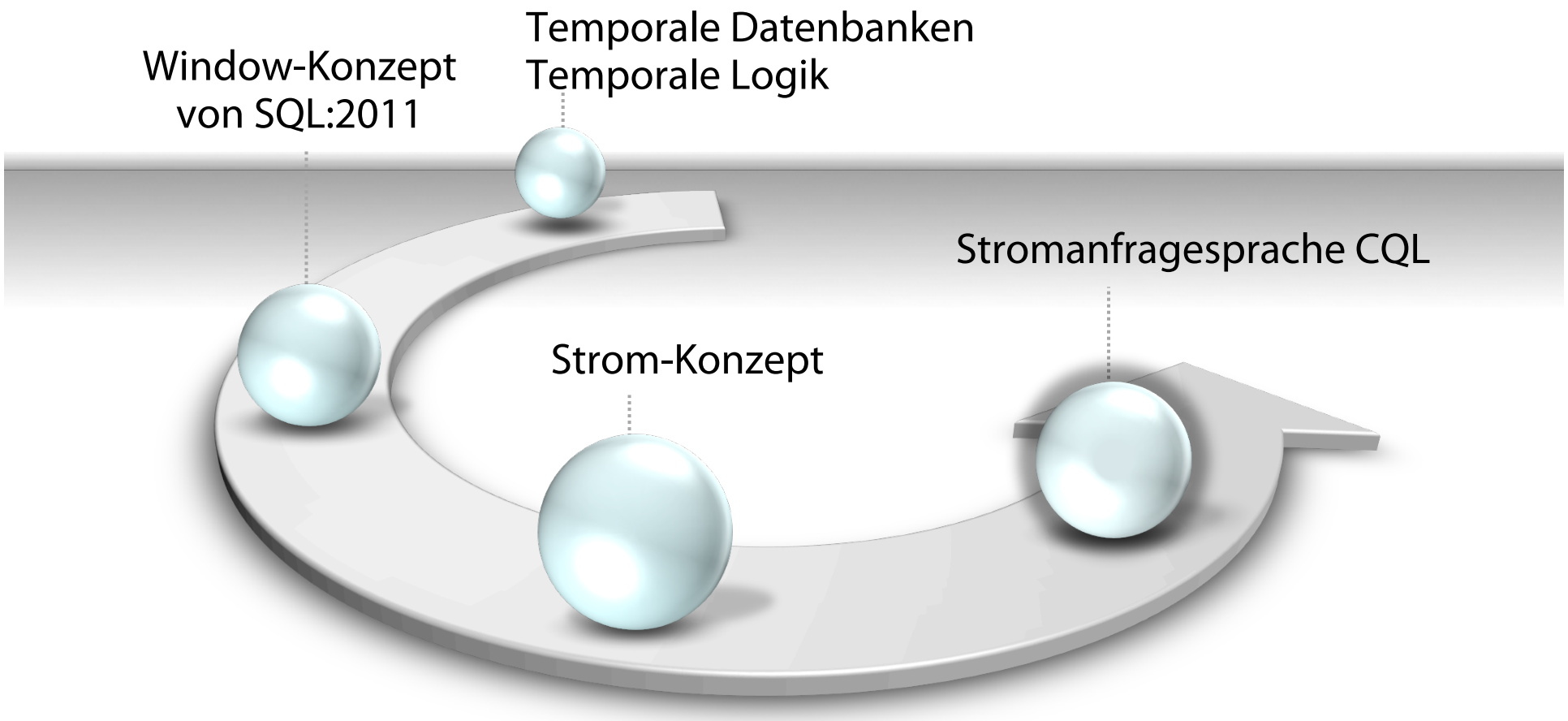
- Transiente Ströme
- Kontinuierliche Anfragen
- Sequentieller Zugriff
- Begrenzter Speicher
- Ankunftsreihenfolge ist entscheidend
- Aktive Datenverarbeitung
- Ggf. multi-GB-Ankunftsraten mit Schwankungen
- Realzeitanforderungen
- Approximationen OK
- Unvorhersagbare variable Ankunftsraten und Datencharakteristiken

Standards?

- Kein Standard in Sicht ☹
- Wir können aber Prinzipien an Beispielsprachen studieren
- Beispiel: Fensterbasierte Systeme

Non-Standard-Datenbanken

Von temporalen Datenbanken zu Stromdatenbanken



STREAM: Stanford Stream Data Manager

- DSMS für Ströme und statische Daten
- Zeitstempel implizit vergeben (Systemzeit)
- Relationale Modellierung ergänzt um Stromkonzept
- Zentralisiertes Servermodell
- CQL: Deklarative Sprache für registrierte kontinuierliche Anfragen über Strömen und statischen Relationen

A. Arasu, S. Babu and J. Widom. The CQL Continuous Query Language: Semantic Foundations and Query Execution, VLDB Journal, **2005**

Konkrete Sprache – CQL

- Relationale Anfragesprache: SQL
- Fensterspezifikationssprache von SQL:2011
  - Tupelbasierte Fenster
  - Zeitbasierte Fenster
  - Partitionierende Fenster
- Einfaches Stichproben-Konstrukt "X% Sample"

CQL Beispielanfrage 1

- Zwei Ströme, sehr einfaches Schema für Beispielzwecke:
  Orders (orderID, customer, cost)
  Fulfillments (orderID, clerk)

- Informationsbedarf natürlichsprachlich ausgedrückt:
  Total cost of orders fulfilled over the last day by clerk "Sue" for customer "Joe"

- Anfrage in CQL:
  Select Sum(O.cost)
  From Orders O**[∞]**, Fulfillments F**[Range 1 Day]**
  Where O.orderID = F.orderID And F.clerk = "Sue"
  And O.customer = "Joe"

CQL Beispielanfrage 2

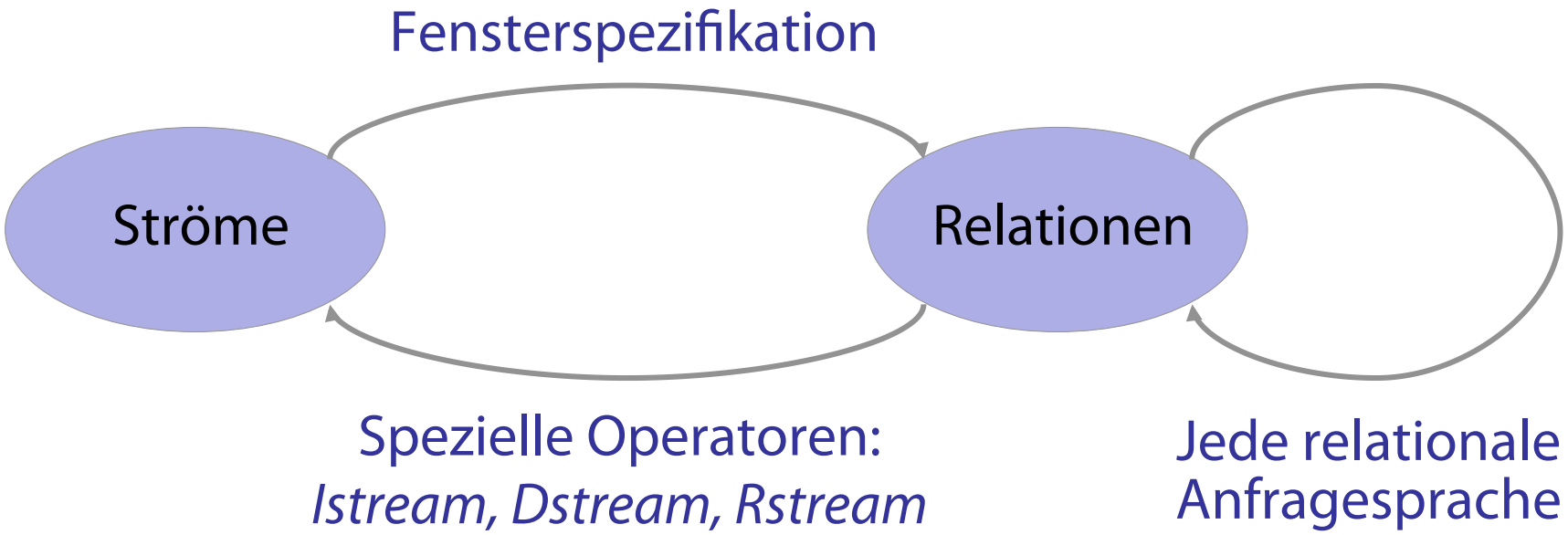
Using a 10% sample of the Fulfillments stream, take the 5 most recent fulfillments for each clerk and return the maximum cost

```
Select F.clerk, Max(O.cost)
From Orders O[∞] ,
    Fulfillments F[Partition By clerk Rows 5] 10% Sample
Where O.orderID = F.orderID
Group By F.clerk
```

Relationen und Ströme

- Annahme: Globale, diskrete, geordnete Menge von Zeitpunkten
- Relation
  - Bildet Zeitpunkte *T* auf Tupelmengen *R* ab
- Strom
  - Menge von (*Tupel, Zeitstempel*)-Elemente
  - Definition with create stream s as select …
  - We write s: select ... for brevity
- Anfragen werden beim DSMS registriert ("kontinuierliche" Anfragen)

Konversion


Fensterspezifikation
Ströme
Relationen
Spezielle Operatoren:
Istream, Dstream, Rstream
Jede relationale
Anfragesprache

Konversion – Definitionen

- Strom-zu-Relation-Operator S[…]
  - $S[W]$ ist eine Relation — zum Zeitpunkt T sind alle Tupel im Fenster W, angewendet auf den Strom S bis zum Zeitpunkt T, enthalten
  - Wenn $W = \infty$, sind all Tupel aus S bis zu T enthalten
  - W definiert Zeitinterval, Anzahl Tupel, und Verschiebung
- Relation-zu-Strom-Operatoren
  - Istream(R) enthält alle (r,T) wobei $r \in R$ zum Zeitpunkt T aber $r \notin R$ zum Zeitpunkt $T-1$
  - Dstream(R) enthält alle (r,T) wobei $r \in R$ zum Zeitpunkt $T-1$ aber $r \notin R$ zum Zeitpunkt T
  - Rstream(R) enthält alle (r,T) wobei $r \in R$ zum Zeitpunkt T

Präzisierung – Multimengensemantik

- Multimenge: Elemente sind Tupel (x, k), wobei $k > 0$ einen Zähler darstellt,
- $Istream(R) := \{ (x, m\text{-}n) \mid$
  $(x, m) \in R$ zum Zeitpunkt T,
  $(x, n) \in R$ zum Zeitpunkt $T\text{-}1$,
  $m\text{-}n > 0 \}$
  $\cup$
  $\{ (x, m) \mid$
  $(x, m) \in R$ zum Zeitpunkt T,
  $(x, n) \notin R$ zum Zeitpunkt $T\text{-}1$ $\}$

Abstrakte Semantik – Beispiel 1

Select F.clerk, Max(O.cost)
From O [∞], F [Rows 1000]
Where O.orderID = F.orderID
Group By F.clerk

Maximum-cost order fulfilled by each clerk in last 1000 fulfillments

Abstrakte Semantik – Beispiel 1

Select F.clerk, Max(O.cost)
From O [∞], F [Rows 1000]
Where O.orderID = F.orderID
Group By F.clerk

- Zum Zeitpunkt *T*: Ganzer Strom *O* und die letzten 1000 Tupel von *F* als Relation
- Evaluiere Anfrage, aktualisiere Ergebnisrelation zum Zeitpunkt *T*

Abstrakte Semantik – Beispiel 1

Select **Istream**(F.clerk, Max(O.cost))
From O [∞], F [Rows 1000]
Where O.orderID = F.orderID
Group By F.clerk

- Zum Zeitpunkt *T*: Ganzer Strom *O* und die letzten 1000 Tupel von *F* als Relation
- Evaluiere Anfrage, aktualisiere Ergebnisrelation zum Zeitpunkt *T*
- **Streamed result:** Neues Element(<clerk,max>,T) wenn<clerk,max> sich in *T–1* ändert

Abstrakte Semantik – Beispiel 2

Relation CurPrice(stock, price)

```
Select stock, Avg(price)
From Istream(CurPrice) [Range 1 Day]
Group By stock
```

Average price over last day for each stock

Abstrakte Semantik – Beispiel 2

Relation CurPrice(stock, price)

Select stock, Avg(price)
From Istream(CurPrice) [Range 1 Day]
Group By stock

Average price over last day for each stock

- *Istream* liefert Historie von *CurPrice*
- Lege Fenster auf Historie, hier genau ein Tag, damit zurück zur Relation, dann gruppieren und aggregieren

Annahmen

- Aktualisierungen von Relationen beinhalten Zeitstempel
- "Gutmütige" Ströme und Relationenänderungen:
  - Ankunft in der richtigen Reihenfolge
  - Keine "Verzögerung", d.h. keine langen Pausen und dann geht's mit der Verarbeitung bei alten Zeitstempeln weiter
- "Missliches" Verhalten separat behandelt, nicht in der Anfragesprache

Einfache Verbundanfrage

Select * From Strm, Rel

Where Strm.A = Rel.B

- Standardfenster [∞] für *Strm*
- Eventuell gewünscht:
  Now-Fenster für strombasierte Verbunde

Select Istream(O.orderID, A.City)
From Orders O, AddressRel A
Where O.custID = A.custID

Einfache Verbundanfrage

Select * From Strm, Rel

Where Strm.A = Rel.B

- Standardfenster [∞] für *Strm*

- Kein Standard:
  Now-Fenster für strombasierte Verbunde

Select Istream(O.orderID, A.City)
From Orders O**[Now]**, AddressRel A
Where O.custID = A.custID

Istream, *Rstream*, Fenster

- Emit 5-second moving average on every timestep

  Select Istream(Avg(A)) From S [Range 5 seconds]

  Hier wird nur ein Ergebnis erzeugt, wenn der Mittelwert sich ändert!

- To emit a result on every timestep

  Select **Rstream**(Avg(A)) From S [Range 5 seconds]

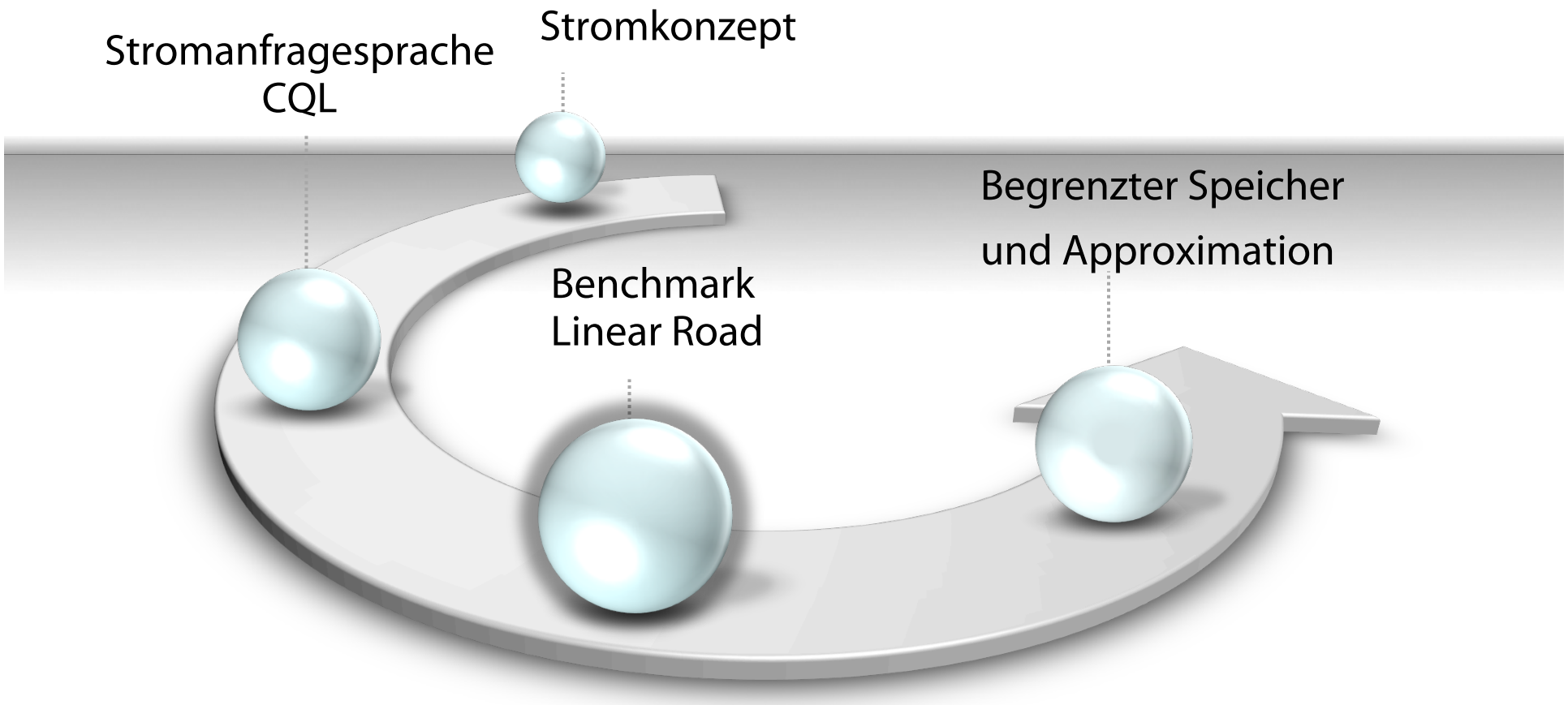
- To emit a result every second

  Select Rstream(Avg(A))

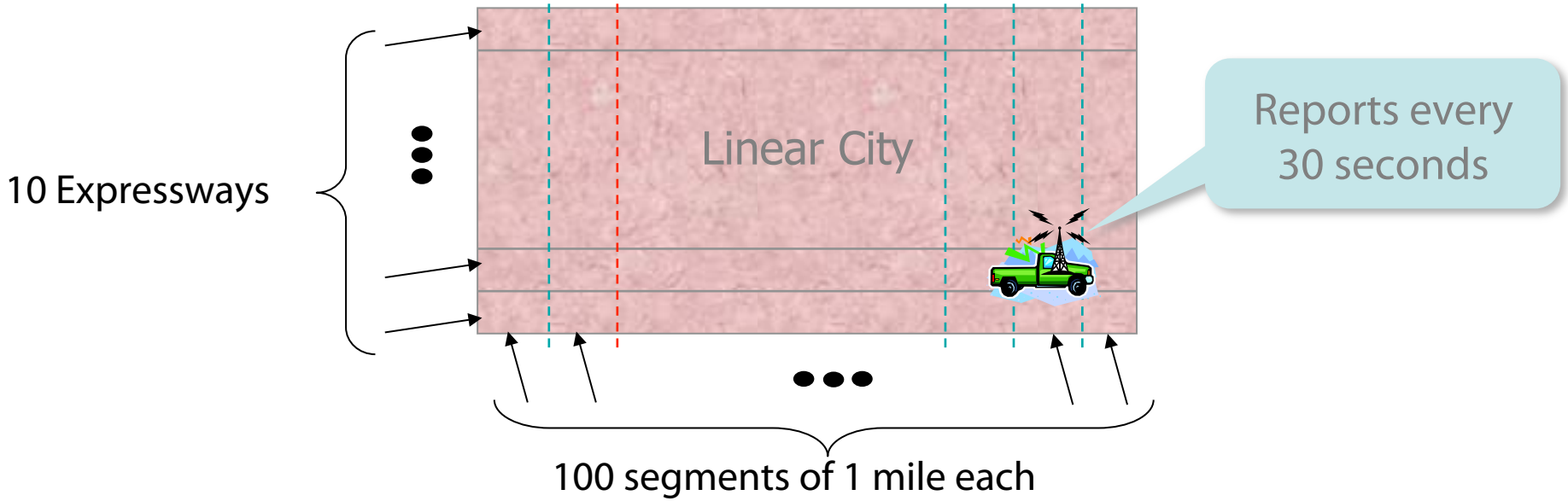
  From S[Range 5 seconds **Slide 1 second**]

Non-Standard-Datenbanken

Von temporalen Datenbanken zu Stromdatenbanken



Benchmark: “Linear Road”



Eingabestrom: Car Locations (CarLocStr)

car_id	speed	exp_way	lane	x_pos
1000	55	5	3 (Right)	12762
1035	30	1	0 (Ramp)	4539
...	...	...	...	...

Linear Road-Benchmark

- Sammlung von kontinuierlichen Anfragen auf realen Verkehrsmanagement-Situationen
- Beispiele:
  - Stream car segments based on x-positions **(leicht)**
  - Identify probable accidents **(mittel)**
  - Compute toll whenever car enters segment **(schwierig)**
- Messlatte: Skalierung auf so viele Expressways wie möglich, ohne in der Verarbeitung zurückzufallen

Linear Road: A Stream Data Management Benchmark, A. Arasu et al., Proceedings of the 30th VLDB Conference, Toronto, Canada, 2004

http://www.cs.brandeis.edu/~linearroad/
http://infolab.stanford.edu/stream/cql-benchmark.html
http://www.it.uu.se/research/group/udbl/lr.html

Einfaches Beispiel

Monitor speed and segments of cars 1-100

Select car_id, speed, x_pos/5280 as segment
From CarLocStr
Where car_id >= 1 and car_id <= 100

+/-	Timestamp	car_id	speed	segment
+	6/6/03 12:34:05	22	23	0
+	6/6/03 12:34:05	10	23	1
+	6/6/03 12:34:05	16	23	11
+	6/6/03 12:34:05	2	30	12
+	6/6/03 12:34:05	25	25	15
+	6/6/03 12:34:05	23	26	18
+	6/6/03 12:34:05	18	20	24
+	6/6/03 12:34:05	5	30	29
+	6/6/03 12:34:05	12	26	40
+	6/6/03 12:34:05	1	27	41
+	6/6/03 12:34:05	4	23	47
+	6/6/03 12:34:05	29	30	53
+	6/6/03 12:34:05	28	30	55
+	6/6/03 12:34:05	7	32	65
+	6/6/03 12:34:05	6	28	99
+	6/6/03 12:34:05	8	30	96
+	6/6/03 12:34:05	9	30	93
+	6/6/03 12:34:05	14	27	90
+	6/6/03 12:34:05	27	27	82
+	6/6/03 12:34:05	26	28	78
+	6/6/03 12:34:05	20	23	77
+	6/6/03 12:34:05	3	23	76
+	6/6/03 12:34:05	21	23	75
+	6/6/03 12:34:05	13	27	71
+	6/6/03 12:34:05	19	32	68

Schwieriges Beispiel

Whenever a car enters a segment, issue it the current toll for that segment

+/-	Timestamp	E.car_id	E.seg	T.toll
+	6/6/03 12:34:35	6	98	8
+	6/6/03 12:34:35	8	95	4
+	6/6/03 12:34:35	9	92	10
+	6/6/03 12:34:35	14	89	7
+	6/6/03 12:34:35	27	81	9
+	6/6/03 12:34:35	26	77	4
+	6/6/03 12:34:35	21	74	9
+	6/6/03 12:34:35	13	70	2
+	6/6/03 12:34:35	19	67	9
+	6/6/03 12:34:35	11	65	5
+	6/6/03 12:34:35	17	60	4
+	6/6/03 12:34:35	24	35	2
+	6/6/03 12:34:36	53	95	5
+	6/6/03 12:34:36	55	91	8
+	6/6/03 12:34:36	45	90	6
+	6/6/03 12:34:36	35	85	10
+	6/6/03 12:34:36	40	79	8
+	6/6/03 12:34:36	780	79	9
+	6/6/03 12:34:36	784	74	10
+	6/6/03 12:34:36	37	73	3
+	6/6/03 12:34:36	46	71	6
+	6/6/03 12:34:36	739	71	7
+	6/6/03 12:34:36	757	67	10
+	6/6/03 12:34:36	776	65	6
+	6/6/03 12:34:36	50	64	7

Maut-Beispiel in CQL

```
Select Rstream(E.car_id, E.seg, T.toll)
From CarSegEntryStr [NOW] as E, SegToll as T
Where E.loc = T.loc
```

CarSegEntryStr: Select Istream(*) From CurCarSeg

CurCarSeg:

```
  Select car_id, x_pos/5280 as seg,
         Location(expr_way, dir, x_pos/5280) as loc
  From CarLocStr [Partition By car_id Rows 1]
```

Maut-Beispiel in CQL (2)

SegToll:

```
Select S.loc, BaseToll * (V.volume – 150)^2
From SegAvgSpeed as S, SegVolume as V
Where S.loc = V.loc and S.avg_speed < 40.0
```

SegAvgSpeed:

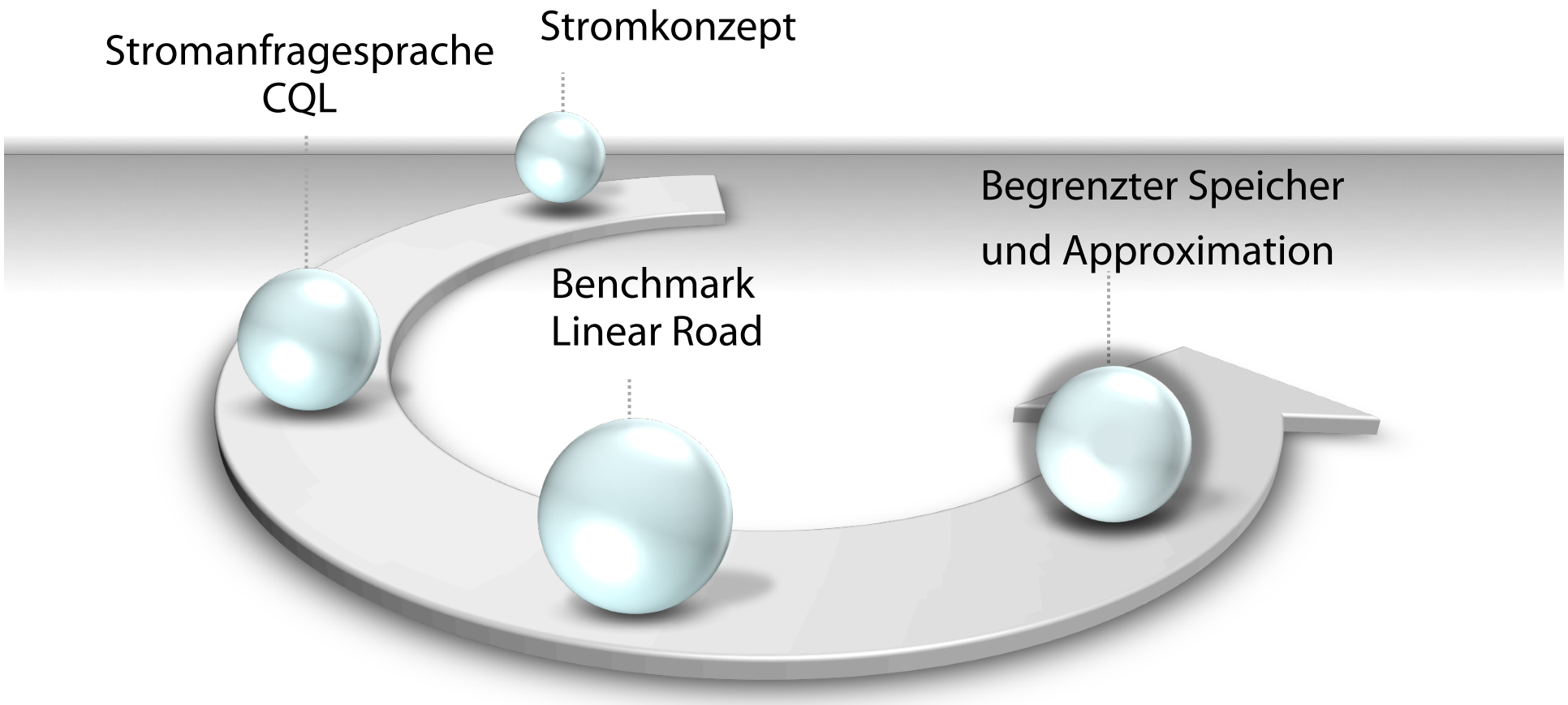
```
Select loc, Avg(speed) as avg_speed
From CarLocStr [Range 5 minutes]
Group By location(expr_way, dir, x_pos/5280) as loc
```

SegVolume:

```
Select loc, Count(*) as volume
From CurCarSeg
Group By loc
```

Non-Standard-Datenbanken

Von temporalen Datenbanken zu Stromdatenbanken



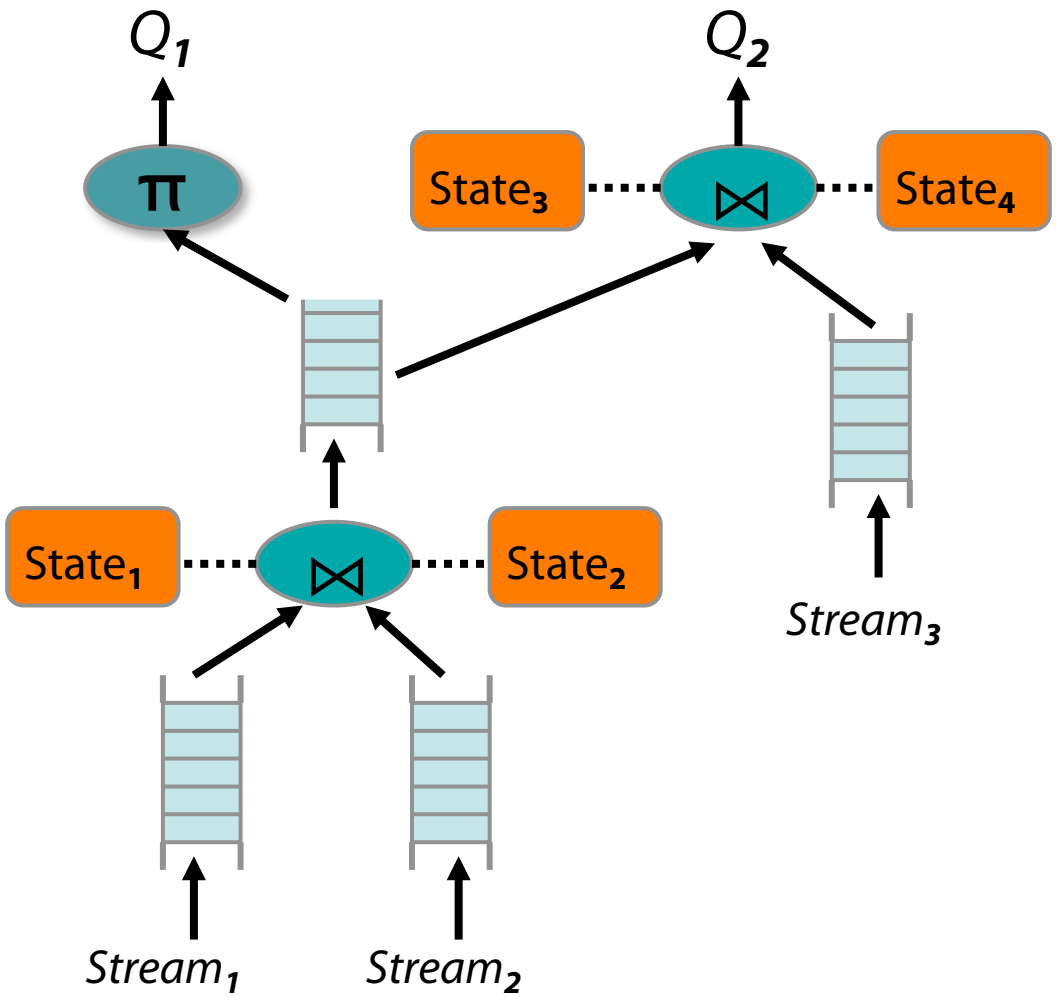
Implementierung eines DSMS – Designeinflüsse

- **Variable Datenraten** mit **hohen Spitzen** mit ...
- ... **kaum vorhersagbarer Verteilung**
- Hohe **Anzahl** registrierter kontinuierlicher Anfragen
- **Designziele** hiermit umzugehen:
  - Multi-Anfrage-Optimierung von Ausführungsplänen
    - Mehrfachverwendung von internen Teilströmen
  - Reoptimierung von Plänen bei Laständerung (Selbstbeobachtung des Systemverhaltens)
  - Lastabhängige, allmähliches Approximation von korrekten Ausgaben (graceful approximation)

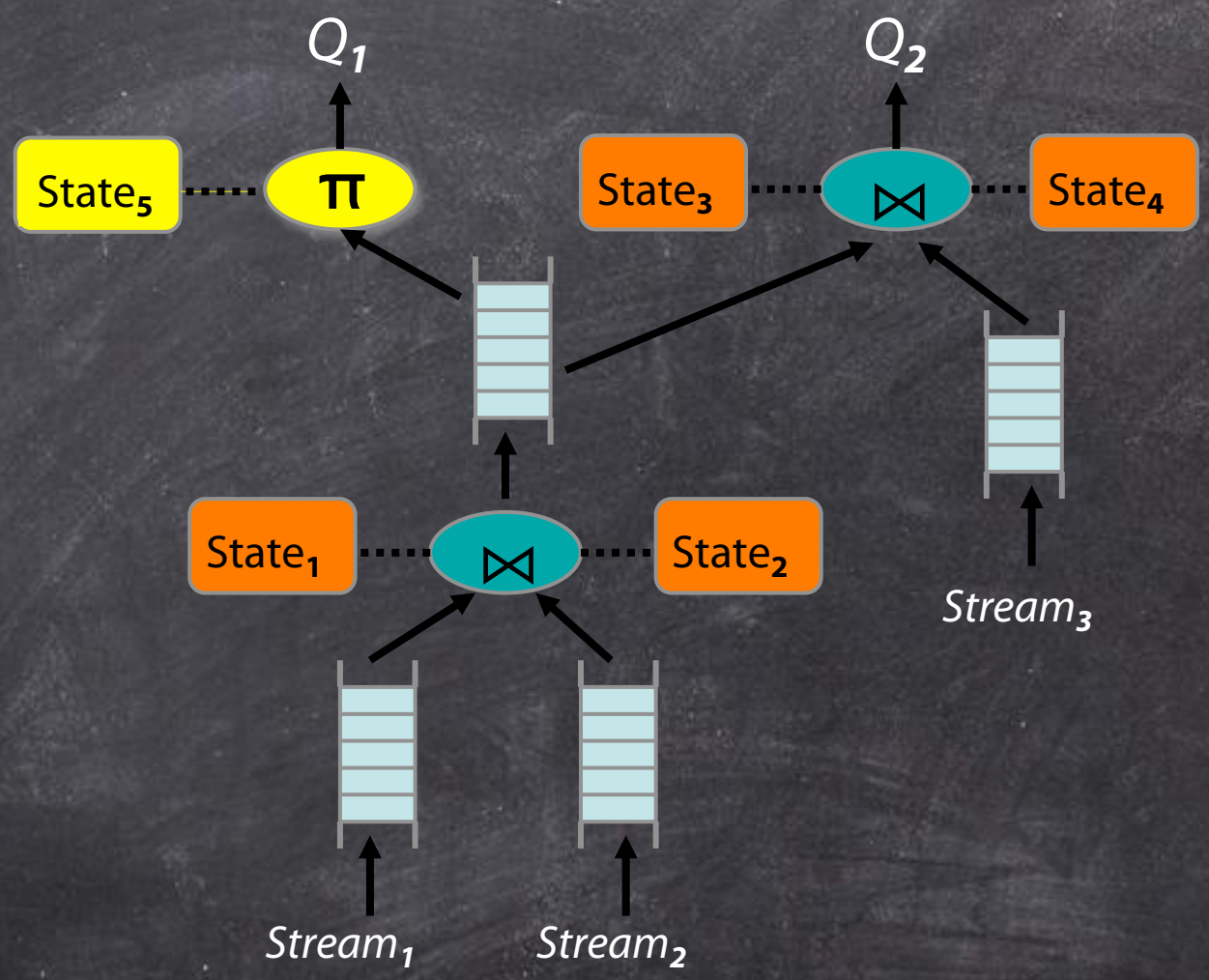
Anfrageverarbeitung

- Registrierung einer kontinuierlichen Query
  → Bestimmung eines Anfrage(bearbeitungs)plans
  - **Wiederverwendung** von existierenden (Teil-)Plänen
- Anfrageplan für Ströme besteht aus:
  - **Operatoren**
    - Selektion
    - Join, Projektion **(ggf. mit IStream)**
    - Sort, Group
  - **Warteschlangen** (Eingabepuffer)
  - **Zuständen** (lokale Speicher für Operatoren)
- Menge von aktiven Plänen durch Verwalter bearbeitet
  - Aufgabe: Versorge Operatoren mit Eingaben

Ein Beispiel



Warum auch Projektion ggf. mit Zustand?

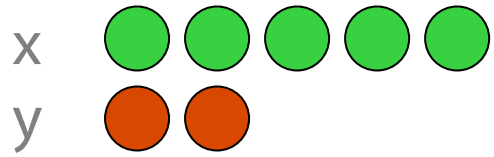


Problembetrachtung

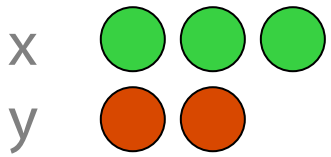
- Fenster sind Möglichkeiten, den Nutzer selbst kleine Einheiten zu definieren zu lassen
- Nicht immer kann man große Fenster (vgl. S[∞]) wegoptimieren
- Auch bei “kleinem” Fenster kann bei einem “Eingabestoß” (burst) eine große Relation entstehen
- → Problematische Verzögerungen
- Auch bei blockierenden Operatoren (z.B. Sort) kann der Speicherbedarf für den jeweiligen Zustand bei großen Relationen sehr groß werden
  - Verarbeitung wird ggf. auch zu langsam

Stromverarbeitungsmodelle – interner Speicher A[·]

- Absolute Werte (Zeitreihenmodell)
  - Jeder neue Wert pro Zeiteinheit x wird gespeichert (x aus Menge von Zeitpunkten)
- Akkumulierte Werte
  - Nur Eingänge (Akkumulation von Zählern)
    - Zähler immer ≥ 0
    - Multimengenmodell
    - (x, y sind Objekte bzw. Objekttypen)

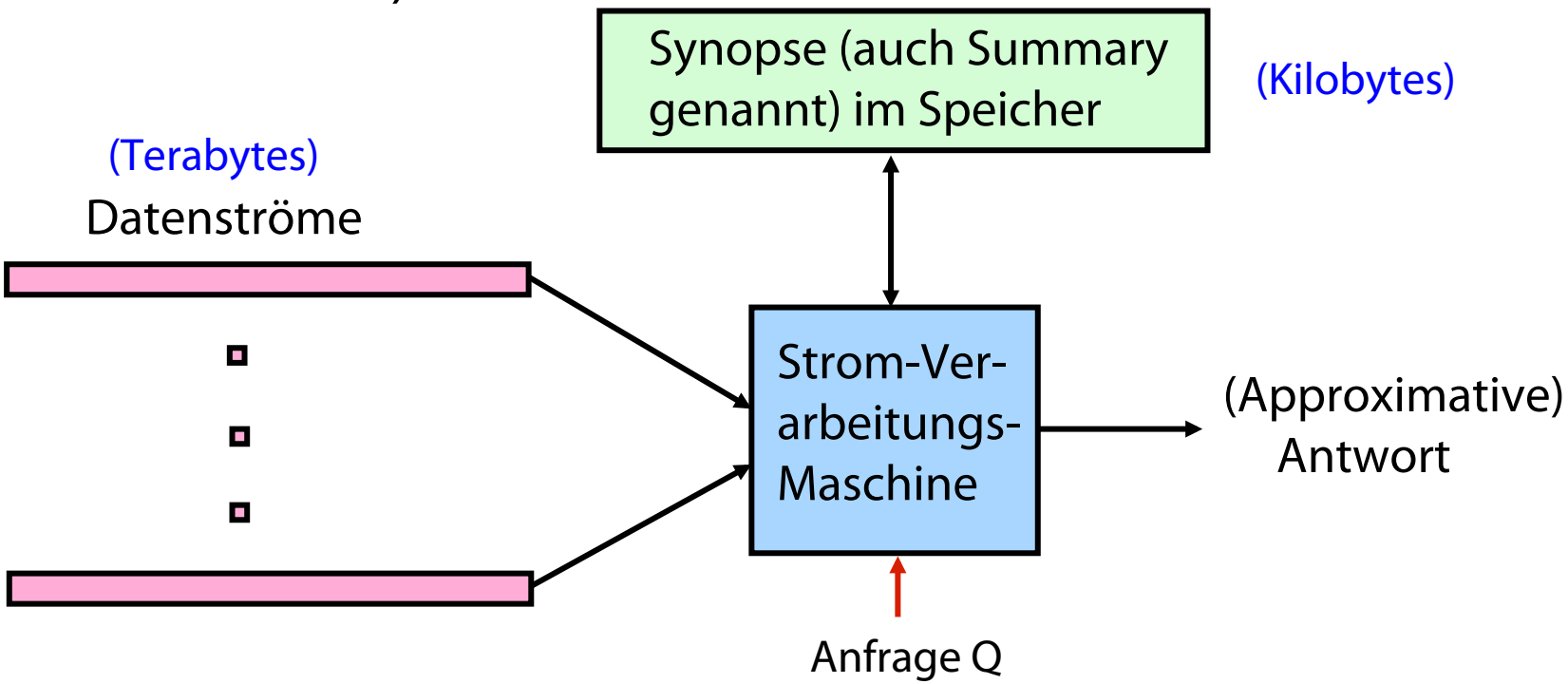


  - Eingänge und Abgänge (Drehkreuzmodell)
    - Multimengenmodell
    - (x, y sind Objekte bzw. Objekttypen)



Approximation

- Daten werden nur einmal betrachtet und
- Speicher für Zustand (stark) begrenzt: O(poly(log(|Strm|)))
- Rechenzeit pro Tupel möglichst klein (auch um Zustand zu modifizieren)



Approximation und Randomisierung

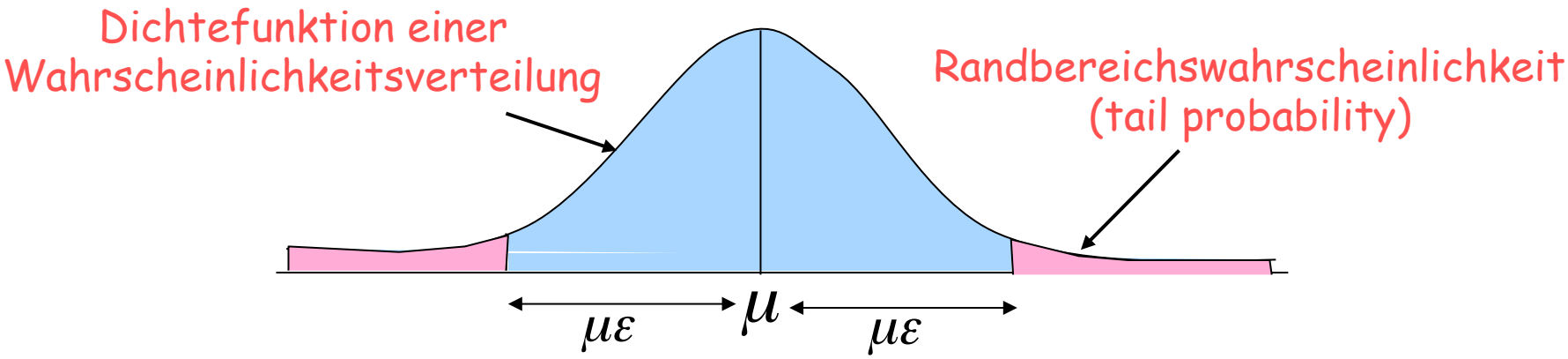
- Viele Probleme sind schwierig, exakt zu berechnen
  - Anzahl der Elemente einer gegebenen Menge von Elementklassen identisch in zwei verschiedenen Strömen?
  - Platzbedarf linear bezogen auf Anzahl der Elementklassen
- **Approximation**: Finde eine Antwort, die korrekt ist bezogen auf einen gegebenen Faktor
  - Finde Antwort im Bereich von ±10% des korrekten Ergebnisses
  - **Genereller**: finde $(1\pm \varepsilon)$-Faktor-Approximation
- **Randomisierung**: Erlaube Fehler mit kleiner Wahrscheinlichkeit
  - Eine von 10000 Antworten darf falsch sein
  - **Genereller**: Erfolgswahrscheinlichkeit $(1-\delta)$
- *Approximation **und** Randomisierung*: (ε, δ)-Approximationen

Probabilistische Garantien

- Benutzerbestimmte *(ε,δ)-Approximationen*
  - Beispiel: Tatsächliche Antworten innerhalb von Faktor 1.1 mit Wahrscheinlichkeit ≥ 0.8 ($\varepsilon = 0.1,\ \delta = 0.2$)
- Wie können wir prüfen, ob durch ein spezielles Verfahren die Gütekriterien eingehalten werden?
  - Verwendung von Randbereichsabschätzungen

Exkurs: Randbereichsabschätzungen

- Generelle Grenzen der Randbereichswahrscheinlichkeit einer Zufallsvariable (Wahrscheinlichkeit, dass Wert weit vom Erwartungswert abweicht)



- <u>Basisungleichungen:</u> Sei X eine Zufallsvariable mit Erwartungswert μ und Varianz Var[X]. Dann gilt für alle $\varepsilon > 0$

Markov:

$$\Pr(X \geq (1+\varepsilon)\mu) \leq \frac{1}{1+\varepsilon}$$

Chebyshev:

$$\Pr(|X-\mu| \geq \mu\varepsilon) \leq \frac{\mathrm{Var}[X]}{\mu^2\varepsilon^2}$$

Stichproben (Abtastung, sampling): Grundlagen

Idee: Ein kleiner Ausschnitt (Stichprobe) *S* einer Datenmenge repräsentiert die Eigenschaften aller Daten

- Zur schnellen Approximierung, wende "modifizierte" Anfrage auf *S* an
- Beispiel: select agg from R where R.e is odd (n=12)

Datenstrom: 9 3 5 2 7 1 6 5 8 4 9 1

Ausschnitt S: 9 5 1 8

- agg = avg → Mittel der ungeraden Elemente in *S* Antwort: 5
- agg = count → Mittel über Summanden abgel. aus e in S mit
  - *n* falls *e* ungerade
  - 0 falls *e* gerade

Antwort: 12*3/4 =9

Randbereichsschätzungen für Summen

- **<u>Hoeffding-Ungleichung</u>**: Seien $X_1, ..., X_m$ unabhängige Zufallsvariablenmit $0 \leq X_i \leq r$.

  Sei $\overline{X} = \frac{1}{m}\sum_i X_i$ und μ der Erwartungswert von $\overline{X}$

  Dann gilt für alle $\varepsilon > 0$

$$\Pr(|\overline{X} - \mu| \geq \varepsilon) \leq 2\exp^{\frac{-2m\varepsilon^2}{r^2}}$$

- Anwendung für Anfragen mit Mittelwert-Operation (avg) :
  - m ist Größe der Untermenge der Stichprobe S, die das Prädikat erfüllt (3 im Beispiel)
  - r ist Bereich der Elemente in der Stichprobe (8 im Beispiel)
- Anwendung auf Zählanfragen (count):
  - m ist Größe der Stichprobe *S* (4 im Beispiel)
  - r ist Anzahl der Elemente n im Strom (12 im Beispiel)

J. M. Hellerstein, P. J. Haas, and H. J. Wang. "Online Aggregation". ACM SIGMOD **1997**

Randbereichsschätzungen für Summen (2)

Für Bernoulli-Versuche (zwei mögliche Erg.) sogar stärke Eingrenzung möglich:

- <u>Chernoff Abschätzung:</u> Sei $X_1, ..., X_m$ unabhängig Zufallsvariable für Bernoulli-Versuche, so dass $Pr[X_i=1] = p$ ($Pr[X_i=0] = 1-p$)
- Sei $X = \sum_i X_i$ und $\mu = mp$ der Erwartungswert von X
- Dann gilt, für jedes $\varepsilon > 0$

$$\Pr(|X - \mu| \geq \mu\varepsilon) \leq 2\exp^{\frac{-\mu\varepsilon^2}{2}}$$

- Verwendung für Anzahlanfrage (count queries):
  - m ist Größe der Stichprobe S (4 im Beispiel)
  - p ist Anteil der ungeraden Elemente im Strom (2/3 im Beispiel)
- Anmerkung: Chernoff-Abschätzung liefert engere Grenzen für Anzahlanfragen als die Hoeffding-Ungleichung

J. M. Hellerstein, P. J. Haas, and H. J. Wang. "Online Aggregation". ACM SIGMOD **1997**

Stichprobem für Datenströme

- **(1)** Betrachtung einer **festgelegten Teilmenge** der Elemente, die im Datenstrom auftreten (z.B. 1 von 10)
- **(2)** Extraktion einer **Teilmenge fester Größe** über einem potentiell unendlichen Strom
  - Zu jedem Zeitpunkt k liegt eine Teilmenge der Größe s vor
  - **Was sind die Eigenschaften der Teilmenge, die verwaltet wird?**
    Für alle Zeitpunkte k, soll jedes der k Elemente, die betrachet wurden, die gleiche Wahrscheinlichkeit haben, in die Teilmenge übernommen zu werden

Verwendung einer festgelegten Teilmenge

- **Szenario:** Auswertung von Anfragen an eine Suchmaschine
  - **Strom von Tupeln:** (user, query, time)
  - **Stromanfrage: How often did a user run the same query in a single day**
  - Ann: Platz für **1/10th** des Stroms
- **Naive Lösung:**
  - Generiere ganzzahlige Zufallszahl **[0..9]** für jede Anfrage
  - Speichere Anfrage falls Zahl = **0**, sonst ignoriere Stromelement (eine Anfrage)

Probleme des naiven Ansatzes

- **Auswertung von: Welcher Bruchteil von Anfragen sind Duplikate?**
  - Nehme an, jeder Benutzer sendet ***x*** Anfragen einmal und ***d*** Anfragen zweimal (insgesamt also ***x+2d*** Anfragen)
    - **Korrekte Antwort: *d*/(*x*+*d*)**
  - **Vorgeschlagene Lösung: Verwende 10% der auftretenden Anfragen aus dem Strom**
    - Stichprobe enthält ***x*/10** der Einfachanfragen und mindestens **2*d*/10** der Mehrfachanfragen
    - Aber es werden nur ***d*/100** der Paare wirklich als Duplikate erkannt
      - **d/100 = 1/10 · 1/10 · d**
    - Von ***d*** "Duplikaten" kommen ***18d/100*** nur genau einmal in der Stichprobe vor
      - **18d/100 = ((1/10 · 9/10)+(9/10 · 1/10)) · d**
  - **Also wäre die Antwort mit der naiven Stichprobe:**
    (d/100)/(x/10) + d/100 + 18d/100 = d/10x + 19d/100

Lösung: Stichprobe mit Benutzern

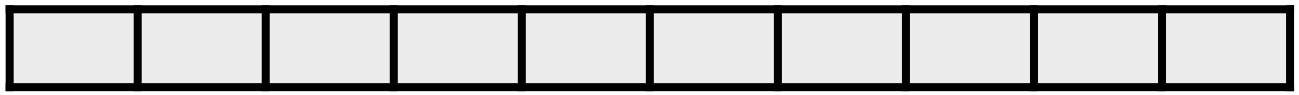
- Wähle **1/10** der **Benutzer** und werte alle ihre Anfagen aus

- Verwende Hash-Funktion, mit der (name, user id) gleichverteilt auf 10 Werte abgebildet wird, speichere Anfragen in Hashtabelle

Generalisierte Lösung

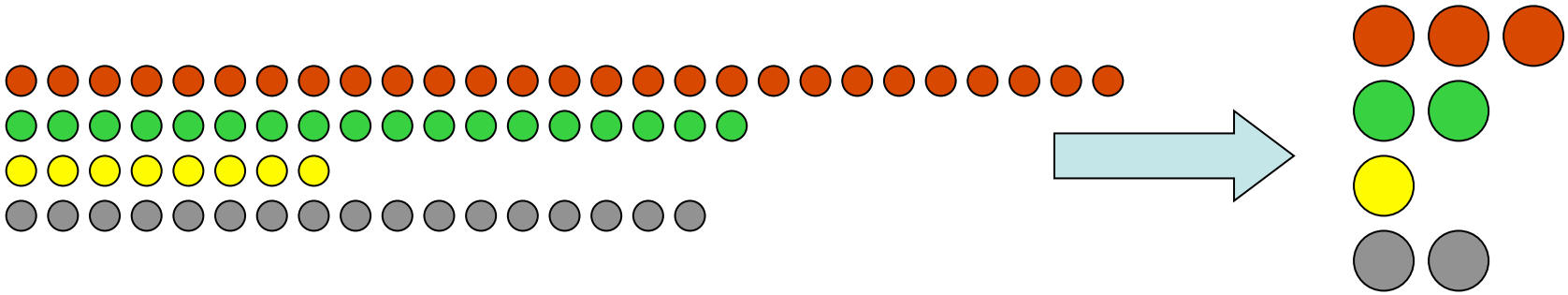
- **Strom von Tupeln mit Schlüsseln:**
  - Schlüssel = Teilmenge der Tupelkomponenten
    - Z.B.: Tupel = (user, search, time); Schlüssel ist **user**
  - Wahl der Schlüssel hängt von der Anwendung ab

- **Bestimmung einer Stichproblem als *a/b* Bruchteil des Stroms:**
  - Hashen der Schlüssel auf ***b*** Hashwerte (Eimer)
  - Wähle das Tupel, falls Hashwert höchstens ***a***



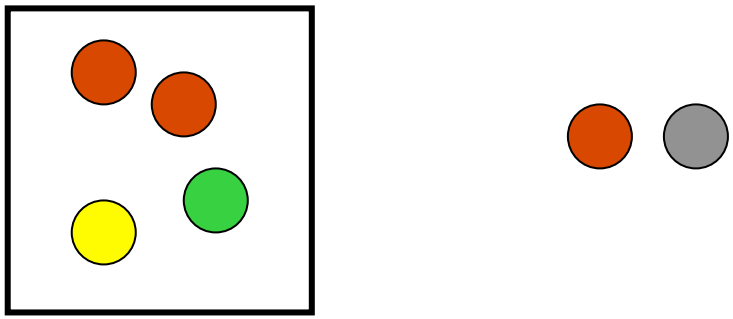
Wie 30% Stichproblem generieren?
Hash auf b=10 Eimer, wähle Tupel, falls in einen der ersten drei Eimer gehasht

Stichproben für Datenströme



- Kernproblem: Wähle Stichprobe von m Elementen gleichverteilt aus dem Strom
- Herausforderung: Länge des Stroms nicht bekannt
  - Wann/wie oft Stichprobe entnehmen?
- Zwei Lösungsvorschläge, für verschiedene Anwendungen:
  - Reservoir Sampling (aus den 80ern?)
  - Min-wise Sampling (aus den 90ern?)

Reservoir Sampling (Reservoir bildet Synopse)



- Übernehme erste m Elemente
- Wähle i-tes Element (i>m) mit Wahrscheinlichkeit m/i
- Wenn neues Element gewählt, ersetze beliebiges Element im Reservoir
- Optimierung: Wenn i groß, berechne jeweils nächstes Element (überspringe Zwischenelemente)

J. S. Vitter. "Random Sampling with a Reservoir". ACM TOMS, **1985**

Reservoir Sampling – Analyse

- Analysieren wir einen einfachen Fall: $m = 1$
- Wahrscheinlichkeit, dass i-tes Element nach n Elementen die Stichprobe bildet:
  - Wahrscheinlichkeit, dass i bei Ankunft gewählt wird × Wahrscheinlichkeit dass i "überlebt"

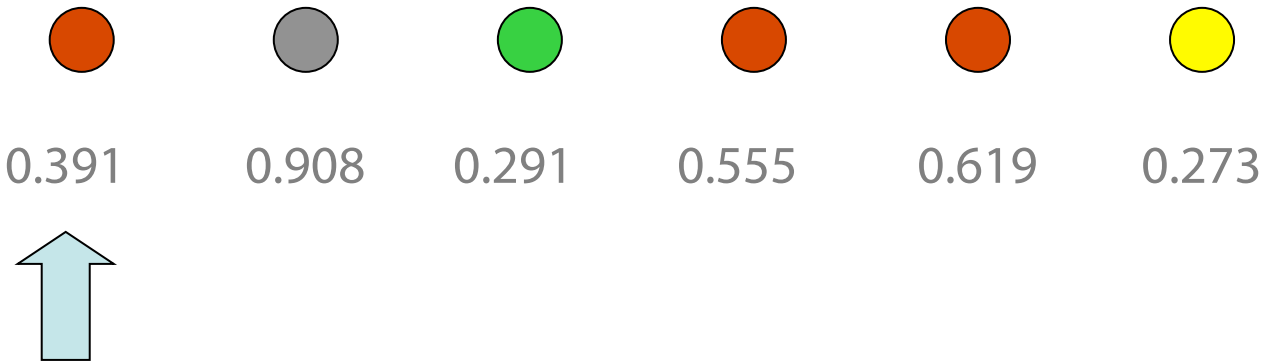
$$\frac{1}{i} \times \left(1 - \frac{1}{i+1}\right) \times \left(1 - \frac{1}{i+2}\right) \ \ldots \ \left(1 - \frac{1}{n-1}\right) \times \left(1 - \frac{1}{n}\right)$$

$$= \frac{1}{\cancel{i}} \times \frac{\cancel{i}}{\cancel{i+1}} \times \frac{\cancel{i+1}}{\cancel{i+2}} \ldots \frac{\cancel{n-2}}{\cancel{n-1}} \times \frac{\cancel{n-1}}{n} = 1/n$$

- Analyse für $m > 1$ ähnlich, Gleichverteilung leicht zu zeigen
- Nachteil: Nicht einfach parallelisierbar

Min-wise Sampling

- Für jedes Element: Wähle Anteil zwischen 0 und 1
- Wähle Element mit kleinstem Anteilswert

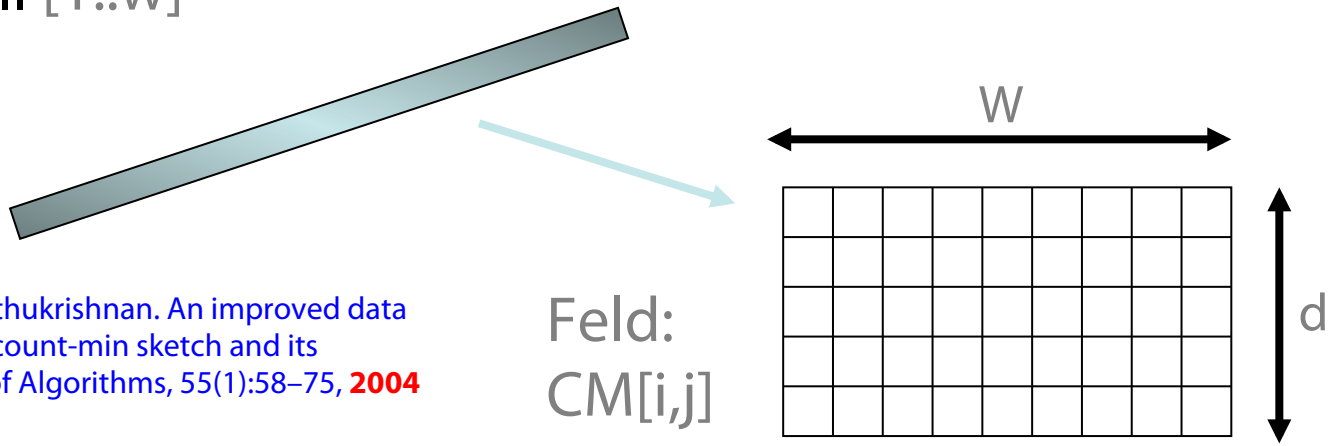


- Jedes Element hat gleiche Chance, kleinstes Element zu werden, also gleichverteilt
- Anwendung auf mehrere Ströme, dann Zusammenführung

S. Nath, P. B. Gibbons, S. Seshan, and Z. R. Anderson. Synopsis diffusion for robust aggregation in sensor networks. In ACM SenSys, **2004**

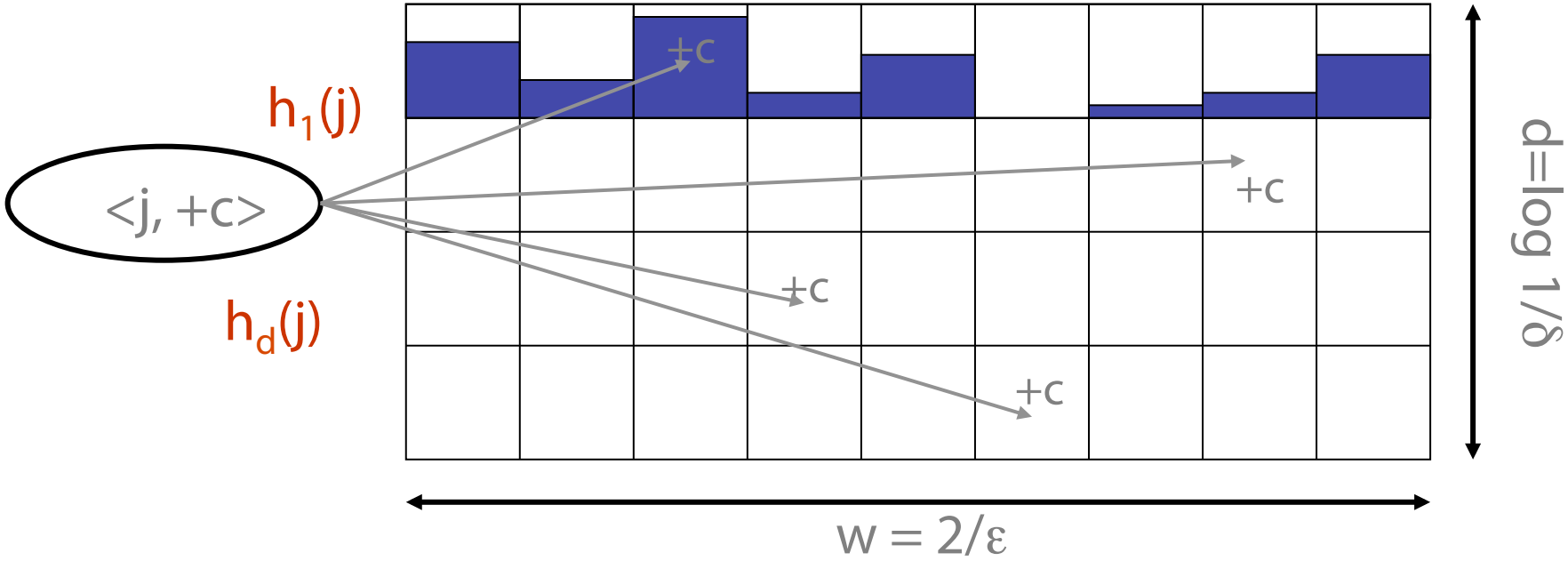
Count-Min-Skizzen (CM-Skizzen)

- Einfache Synopse (genannt Skizze), Basis für viele Strom-Untersuchungsaufgaben (stream mining tasks)
  - hauptsächlich für "Anzahl Elemente pro Typ" (item frequencies)
  - für Zählerakkumulierung und Drehkreuzmodell
- Eingabestrom intern als Vektor A der Dimension N repräsentiert
- Skizze als kleines Feld CM der Größe $w \times d$ dargestellt
- Verwende d Hashfunktionen zur Abbildung der A-Elemente auf Intervall [1..w]



G. Cormode and S. Muthukrishnan. An improved data stream summary: The count-min sketch and its applications. Journal of Algorithms, 55(1):58–75, **2004**

CM-Skizzen – Struktur



- Jedes A[] wird auf CM-Eintrag abgebildet
  - h()' s *paarweise unabhängig*
- Schätze A[j] über den Ausdruck $\min_k \{ CM[k,h_k(j)] \}$
- Parallelisierung: Verschmelzung eintragsweise durch Summierung verschiedener CM-Matrizen möglich

CM-Skizzen – Garantien

- CM-Approximierungsfehler bei Punktanfragen kleiner als $\varepsilon||A||_1$ mit Platzbedarf $O(1/\varepsilon \log 1/\delta)$ *[Cormode, Muthukrishnan'04]*
  - Wahrscheinlichkeit eines größeren Fehlers kleiner als $1-\delta$
  - Ähnliche Garantien für Bereichsanfragen, Quantile, Verbundgrößen (join size), …

- Hinweise
  - Zähler überschätzen durch Hash-Kollisionen
    - Wie begrenzbar in jedem Feld?
  - Nutze Unabhängigkeit über Zeilen, um Konfidenz für min{} Schätzung zu erhöhen

Herleitung der Abschätzung A‘[j] für A[j]

$$A'[j] = \min_k \{ CM[k, h_k(j)] \}$$

Analyse der k-ten Zeile

$$CM[k, h_k(j)] = A[j] + X_{k,j}$$

$$X_{k,j} := \sum_{h_k(i) = h_k(j)} A[i]$$

$$E[X_{k,j}] = \sum_i A[i] \cdot Pr(h_k(i) = h_k(j))$$

$$Pr(h_k(i) = h_k(j)) \leq \frac{1}{w} = \varepsilon/2 \qquad (h \text{ paarw. unabh.})$$

Herleitung der Abschätzung A‘[j] für A[j] (2)

$$E[X_{k,j}] \leq \frac{\varepsilon}{2} \sum_i A[i] = \frac{\varepsilon}{2} \|A\|_1$$

Fehler wenn $X_{k,j} \geq \varepsilon \|A\|_1$

$$P[X_{k,j} \geq \varepsilon \|A\|_1] = Pr[X_{k,j} \geq 2E[X_{k,j}]] \leq \frac{1}{2}$$

Markov-Abschätzung: $Pr(X \geq (1+\varepsilon)\mu) \leq \frac{1}{1+\varepsilon}$

$(1+\varepsilon)\mu$

1

$$Pr[\forall k.\ X_{k,j} \geq \varepsilon \|A\|_1] \leq \underbrace{\frac{1}{2} \cdot \frac{1}{2} \cdot \ldots \frac{1}{2}}_{d} = \frac{1}{2^d} = \frac{1}{2^{\log \frac{1}{\delta}}} = \delta$$

CM-Skizzen – Analyse

Schätze $A'[j] = \min_k \{ CM[k,h_k(j)] \}$

- Analyse: In k'ter Zeile, $CM[k,h_k(j)] = A[j] + X_{k,j}$
  - $X_{k,j} = \Sigma A[i] \mid h_k(i) = h_k(j)$
  - $E[X_{k,j}] \quad = \Sigma A[i]*Pr[h_k(i)=h_k(j)]$
    $\leq (\varepsilon/2) * \Sigma A[i] = \varepsilon \, ||A||_1/2$ (paarweise Unabh. von h)
  - $Pr[X_{k,j} \geq \varepsilon||A||_1] = Pr[X_{k,j} \geq 2E[X_{k,j}]] \leq 1/2$ über Markov Ungl.
- Also, $Pr[A'[j] \geq A[j] + \varepsilon \, ||A||_1] = Pr[\forall k. X_{k,j} > \varepsilon \, ||A||_1] \leq 1/2^{\log 1/\delta} = \delta$
- Endergebnis: $A[j] \leq A'[j]$ und
  mit Wahrscheinlichkeit $1-\delta$ gilt $A'[j] < A[j] + \varepsilon \, ||A||_1$

Zählen der Anzahl der verschiedenen Element

- **Problem:**
  - Datenstrom enthält Elemente aus Grundmenge der Größe ***N***
  - Gesucht ist Anzahl der verschiedenen Elemente in einem gesamten bisheringen Strom zu einem Zeitpunkt, in dem Fenster ausgewertet wird

- **Naiver Ansatz:**
  Speichere gesehene Elemente in Hashtabelle also Synopse

Anzahl **verschiedener** Werte abschätzen

- Aufgabe: Finde Anzahl der verschiedenen Werte in einem Strom von Werten aus [1,...,N] (`count distinct`)
  - Statistik: Anzahl von Arten oder Klassen in Population
  - Informatik: Anfrageoptimierung
  - *Netzwerkbeobachtung:* IP-Adressen mit unterschiedlichem Ziel, Quelle/Ziel-Paare, angeforderte URLs usw

- Beispiel (N=64)

Datenstrom: 3 2 5 3 2 1 7 5 1 2 3 7

Anzahl der verschiedene Werte: 5

- Naiver Ansatz: Hash-Tabelle für alle gesehenen Elemente
- Schwierig auch für CM (gedacht für Multimengen)

Flajolet-Martin Approach

- Wähle Hashfunktion ***h*** zur Abbildung von ***N*** Elementen auf mindestens $\mathbf{log_2} \boldsymbol{N}$ Bits

- Für jedes Stromelement ***a***, sei ***r(a)*** die Anzahl der Nullen am Ende von ***h(a)***
  - **r(a)** = Position der ersten 1 von rechts
    - Z.B. sei ***h(a) = 12***, dann ist ***12*** gleich ***1100*** binär, also ***r(a) = 2***
- Speichere ***R* = maximales *r*(*a*) bisher**
  - $\mathbf{R = max_a\ r(a)}$, über alle Stromelemente ***a*** bisher

- **Geschätzte Anzahl unterschiedlicher Elemente: 2^R**

P. Flajolet and G. N. Martin. Probabilistic counting. In IEEE Conference on Foundations of Computer Science, pages 76–82, 1983. Journal version in Journal of Computer and System Sciences, 31:182–209, **1985**.

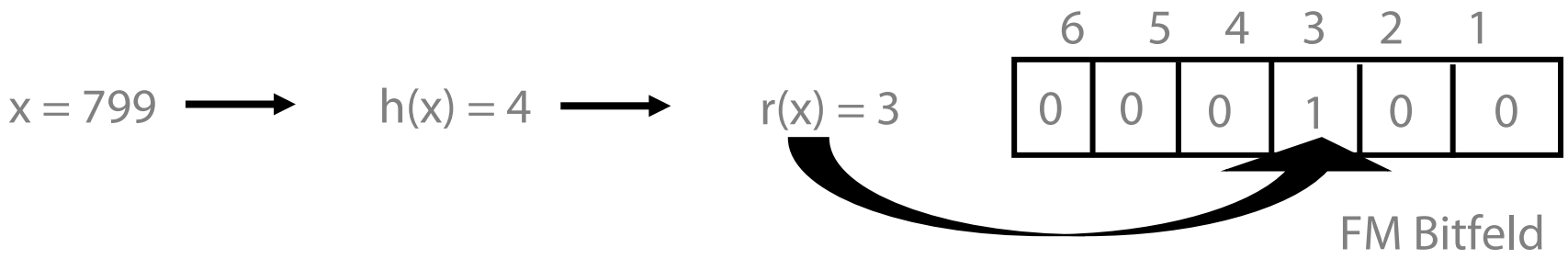
Warum funktioniert das? Intuition

- **h(a)** bildet **a** miz **gleicher Wahrscheinlichkeit** auf jeden von **N** möglichen Werten ab
- Dann ist **h(a)** eine Sequenz von $\mathbf{log_2 N}$ Bits, wobei $\mathbf{2^{-r}}$ der Anteil aller **a**s mit r Nullen am Ende ist
  - Ca. 50% der **a**s hashen auf ***0
  - Ca. 25% der **a**s hashen auf **00
  - Wenn also **r(a)=2** hinten stehen (i.e., Hash ergbit ***100**) dann haben wir wahrscheinlich **ca. 4** verschiedene Element gesehen
- **Also braucht es ein Hash auf 2^r Element bevor man eines mit 0-Suffix der Länge r sieht**

J. Leskovec, A. Rajaraman, J. Ullman:
Mining of Massive Datasets,
http://www.mmds.org

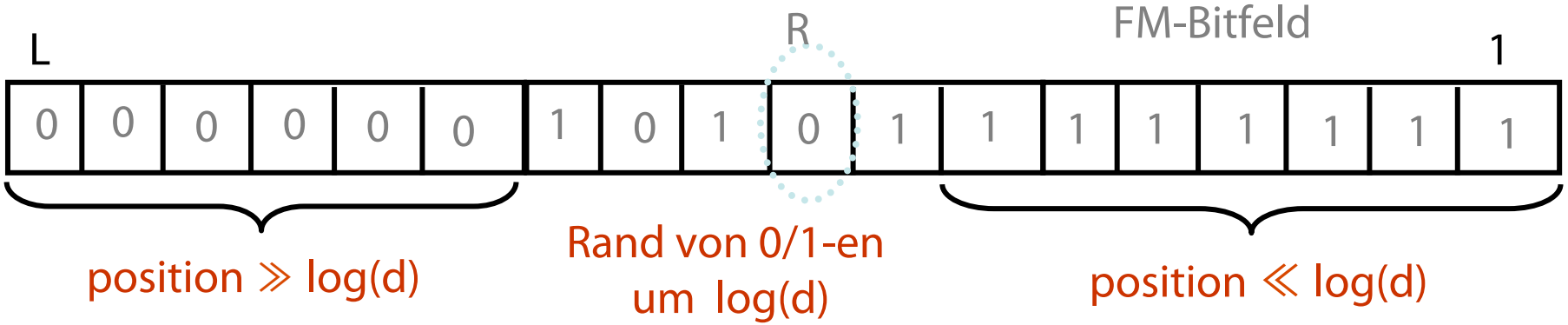
FM-Skizzen [Flajolet, Martin' 85]

- Verwende Hashfunktion zur Abbildung von Eingabeelementen auf i mit Wahrscheinlichkeit 2^{-i}
  - also Pr[h(x) = 1] = ½, Pr[h(x) = 2] = ¼, Pr[h(x)=3] = 1/8 …
  - Konstruiere h() aus gleichverteilter Hashfunktion an anschließendendem "Zählen der Nullen am Ende" durch r
- Aufbau FM-Skizze= Bitfeld von L = log N Bits
  - Initialisiere Bitfelder of 0
  - Für jeden neuen Wert x, setze FM[r(x)] = 1



FM-Skizzen – Analyse

- Bei d verschiedenen Werten, erwarte Abbildung von d/2 Werten nach FM[1], d/4 nach FM[2]…



  - Sei R = Position der rechtesten 0 in FM, Indikator für log(d)
  - [FM85] zeigen, dass E[R] $= \log(\phi d)$, mit $\phi = .7735$
  - Schätzung $d = c2^R$ für Skalierungskonstante $c \approx 1.3$
  - Mittelung mit verschiedenen Hashfunktionen dient zur Verbesserung des Ergebnisses
  - Wieviele Hashfunktionen für best. Anforderung benötigt?

Herleitung der Schätzung von d

$$E(R) = \log(\Phi d) \qquad [FM85]$$

$$= \log \Phi + \log d$$

$$E[R] - \log \Phi = \log d$$

$$\log 2^{E[R]} - \log \Phi = \log d$$

$$\log \frac{2^{E[R]}}{\Phi} = \log d$$

$$d \approx 2^R / \Phi \approx c\, 2^R \qquad c \approx 1.3..$$

FM-Skizzen – Eigenschaften

- Mit $O(1/\varepsilon^2 \log 1/\delta)$ Hashfunktionen, $(1\pm\varepsilon)$ Genauigkeit mit Wahrscheinlichkeit mindestens $1-\delta$
  Ohne Beweis: siehe [Bar-Yossef et al' 02], [Ganguly et al.' 04]
  - 10 Funktionen ≈ 30% Fehler, 100 Funkt. < 10% Fehler

- *Bei Löschung:* Verwende Zähler statt Bits
  - +1 für Einfügungen, -1 für Löschungen
- *Komposition:* komponentenweise OR bzw. +

6	5	4	3	2	1
0	0	1	0	1	1

+

6	5	4	3	2	1
0	1	1	0	0	1

=

6	5	4	3	2	1
0	1	1	0	1	1

  - Schätze $|S_1 \cup \cdots \cup S_k|$ = *Kardinalität der Vereinigungsmenge*

Stichproben und Skizzierung: Zusammenfassung

- Zentrale Idee für viele Stromanalyseverfahren
  - Momente/Verbundaggregate, Histogramme, top-k, Meistfrequentierte Elemente, andere Analyseprobleme, …
- Stichproben eher generelle Repräsentation eines Datensatzes
  - Einfache Stichproben (sampling) nicht für Strommodelle mit Ein- und Abgängen (Drehkreuzmodell) geeignet (es gibt auch hier neue Arbeiten)
- Skizzierung eher für speziellen Zweck
  - FM-Skizze für "Anzahl der Typen",
  - CM-Skizze für "Anzahl Elemente pro Typ" (auch: Verbundgröße bzw. Momentenschätzung …)

Zusammenfassung

Von Stromdatenbanken zu probabilistischen Datenbanken