
Non-Standard-Datenbanken

Stromdatenbanken

Prof. Dr. Ralf Möller

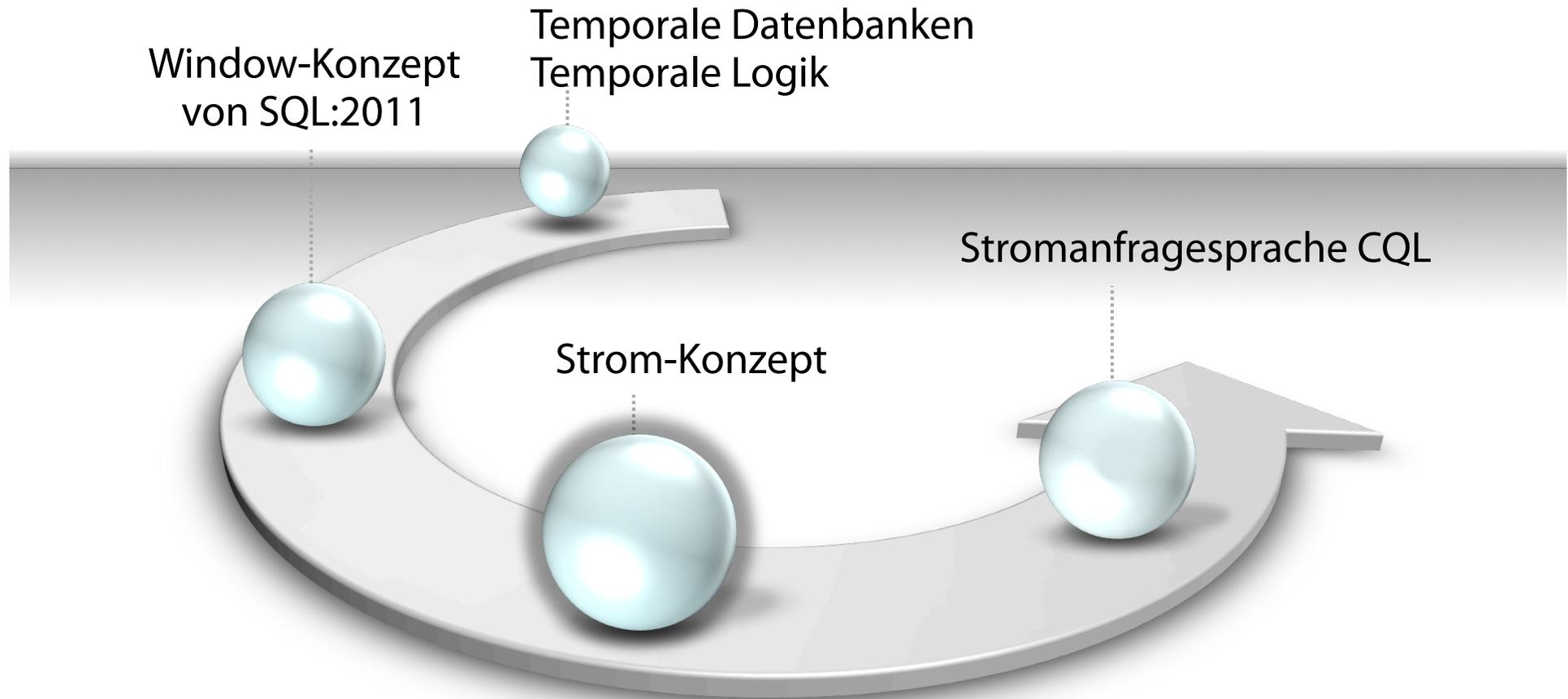
Universität zu Lübeck

Institut für Informationssysteme



Non-Standard-Datenbanken

Von temporalen Datenbanken zu Stromdatenbanken



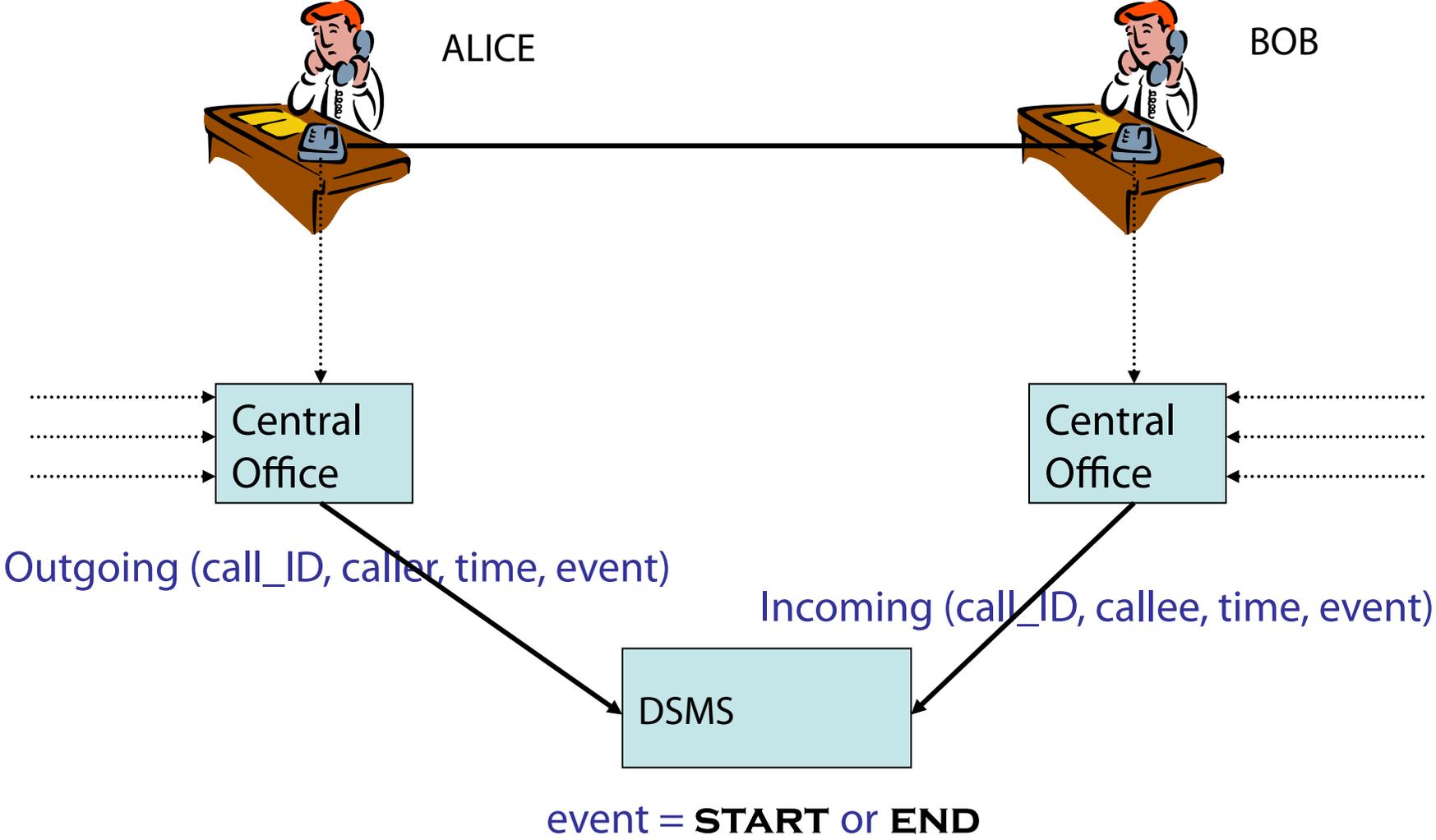
Datenströme: Motivation

- **Traditionelle DBMS** – Daten in endlichen persistenten Dateneinheiten gespeichert (Tabellen, XML-Dokumenten, ...) 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, ...
 - Komm. Anbieter: Gigaspaces, ...

Beispiel 1: Telefondatenauswertung



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

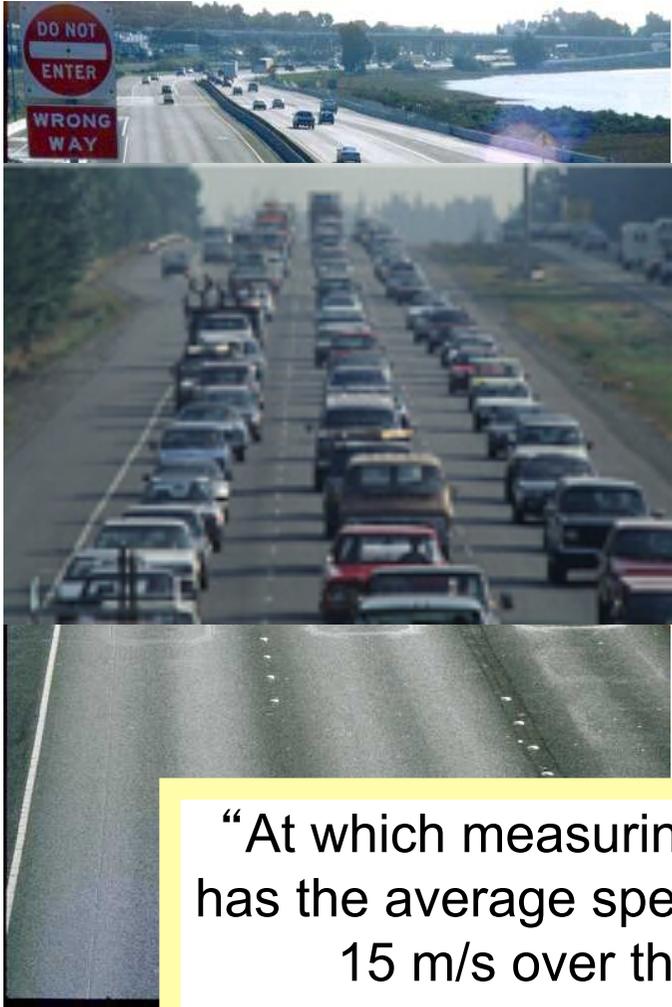
Anfrage 3 (Gruppierung und Aggregation)

- 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

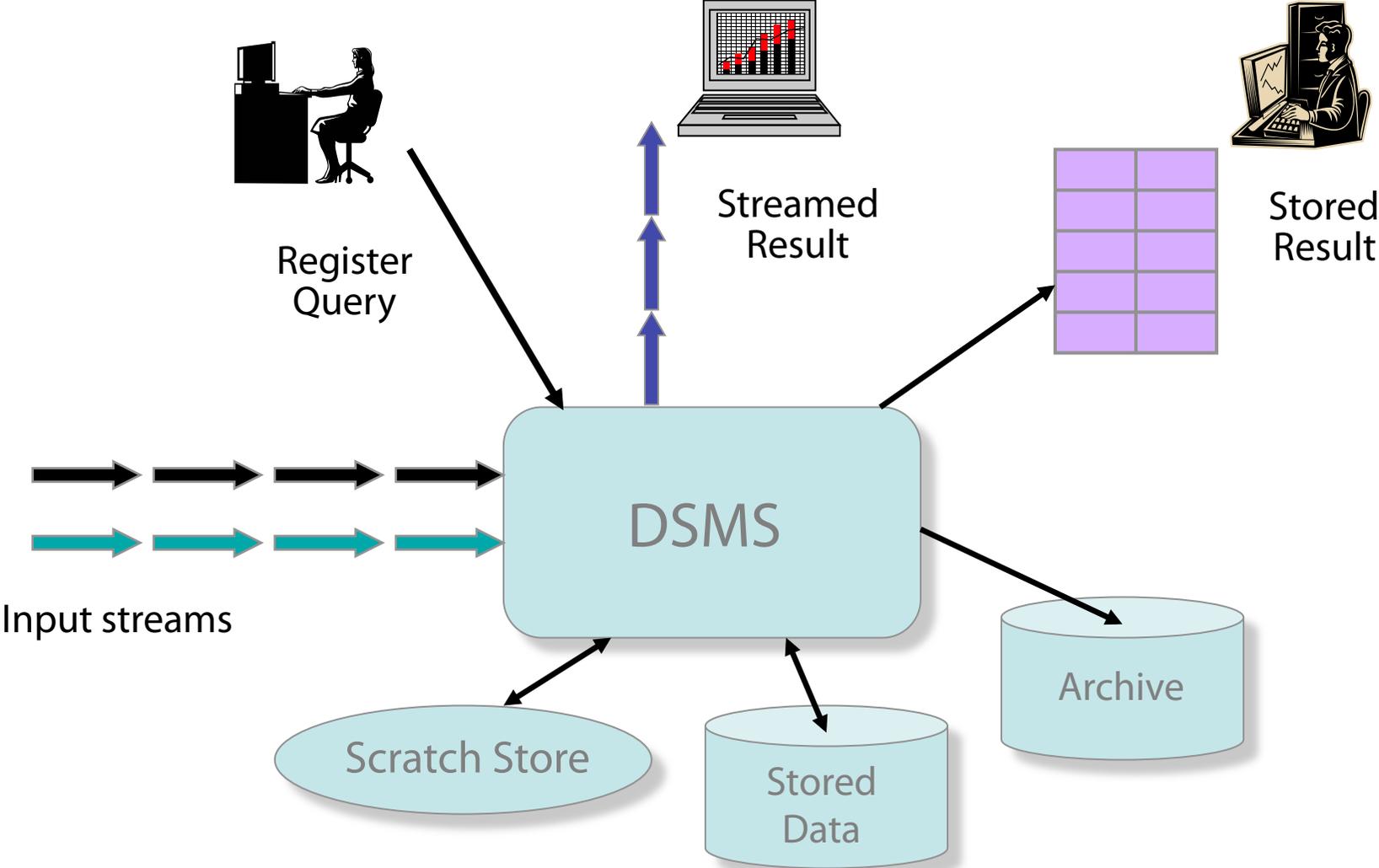


Anforderungen

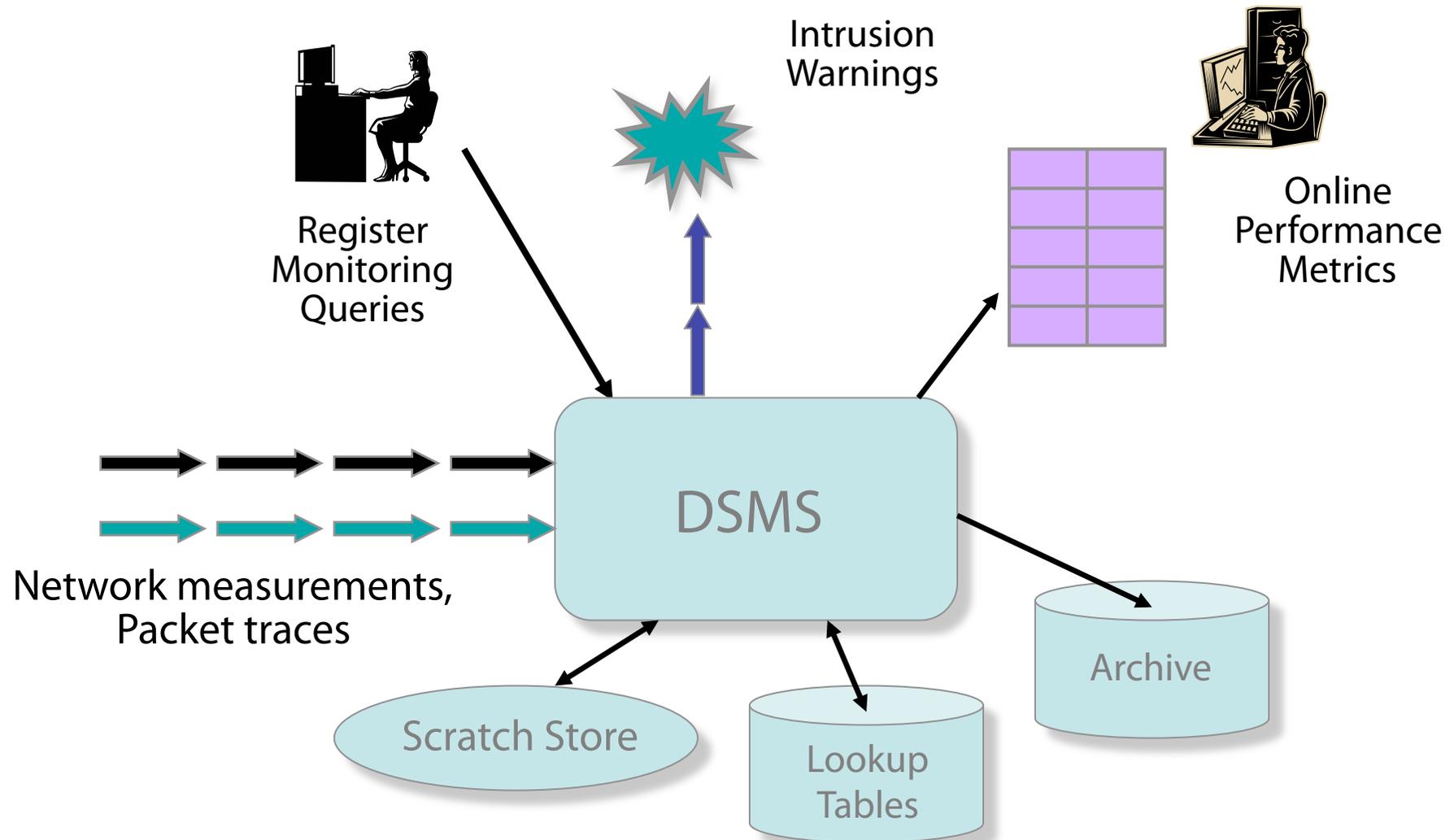
- Deklarative Anfragesprache
- Ausdrucksstark wie (temporales) SQL
 - Verbund von Datenströmen bezogen auf die Zeit
 - Kombination von Datenströmen mit persistenten Datenbasen
 - Anfrageergebnisse als neue Datenströme nutzbar
- Publish/Subscribe Paradigma
 - Subscribe: Nutzer registrieren Anfragen
 - 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

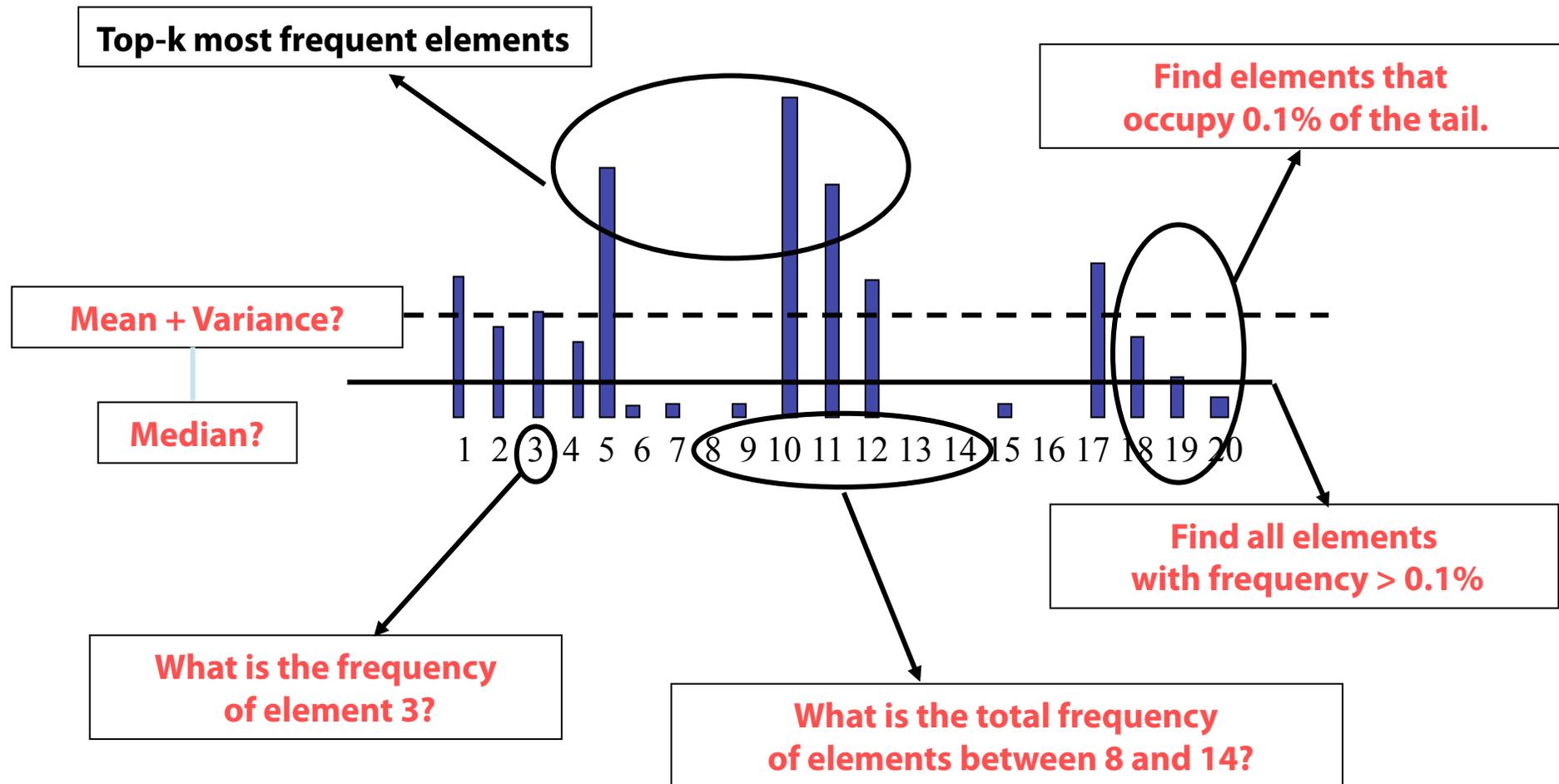


Spezielle Ausprägung: Netzwerküberwachung



Beispiel 3: Auswertung von Netzwerkdaten

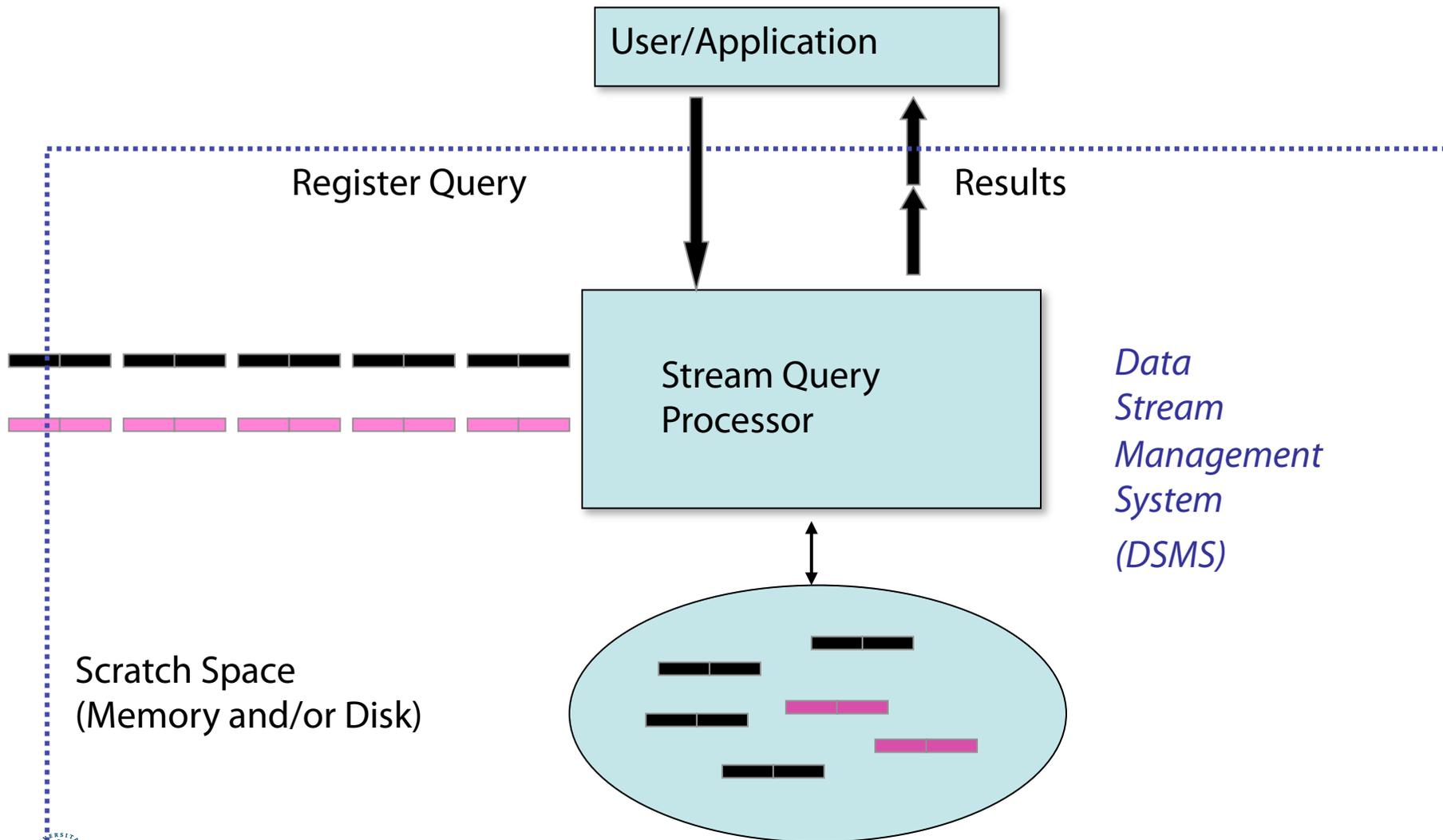
Analytics on Packet Headers – IP Addresses



Rajeev Motwani 2003

How many elements have non-zero frequency?

Data Stream Management System



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-Ankunftsrate mit Schwankungen
- Realzeitanforderungen
- Approximationen OK
- Unvorhersagbare variable Ankunftsrate und Datencharakteristiken



Fensterkonzept

Kein Einfluss von "alten" Daten auf Ergebnis

→ Verschiebbare zeitliche Fenster

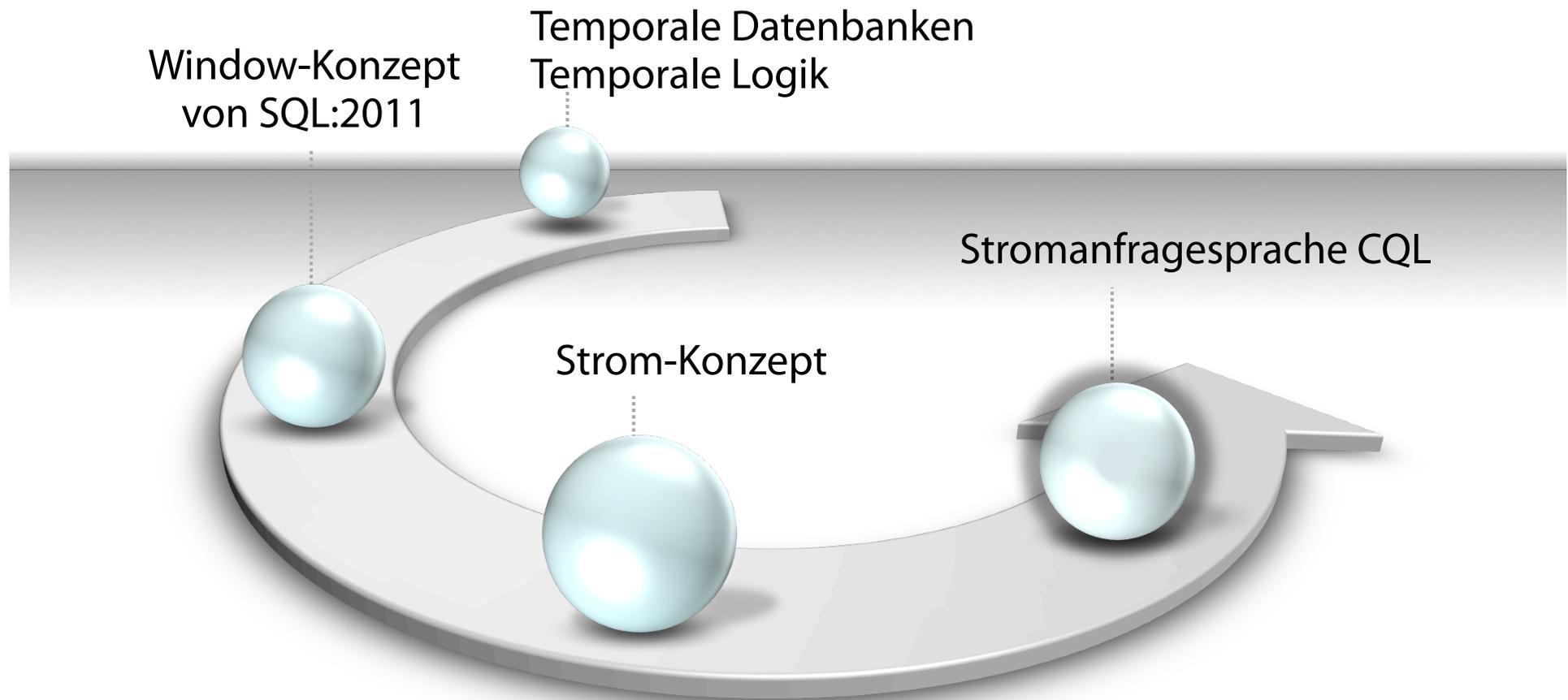
- Endliche Teilsequenzen eines unendlichen Stroms
- Anfragebeantwortung auf neueste Daten fokussiert
- Wichtig für **ausdrucksstarke Anfragen** und deren **effiziente Verarbeitung**

• 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)

Non-Standard-Datenbanken

Von temporalen Datenbanken zu Stromdatenbanken



Standards?

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

STREAM: Stanford Stream Data Manager

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

CQL Beispielanfrage 1

- Zwei Ströme, sehr einfaches Schema für Beispielszwecke:
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



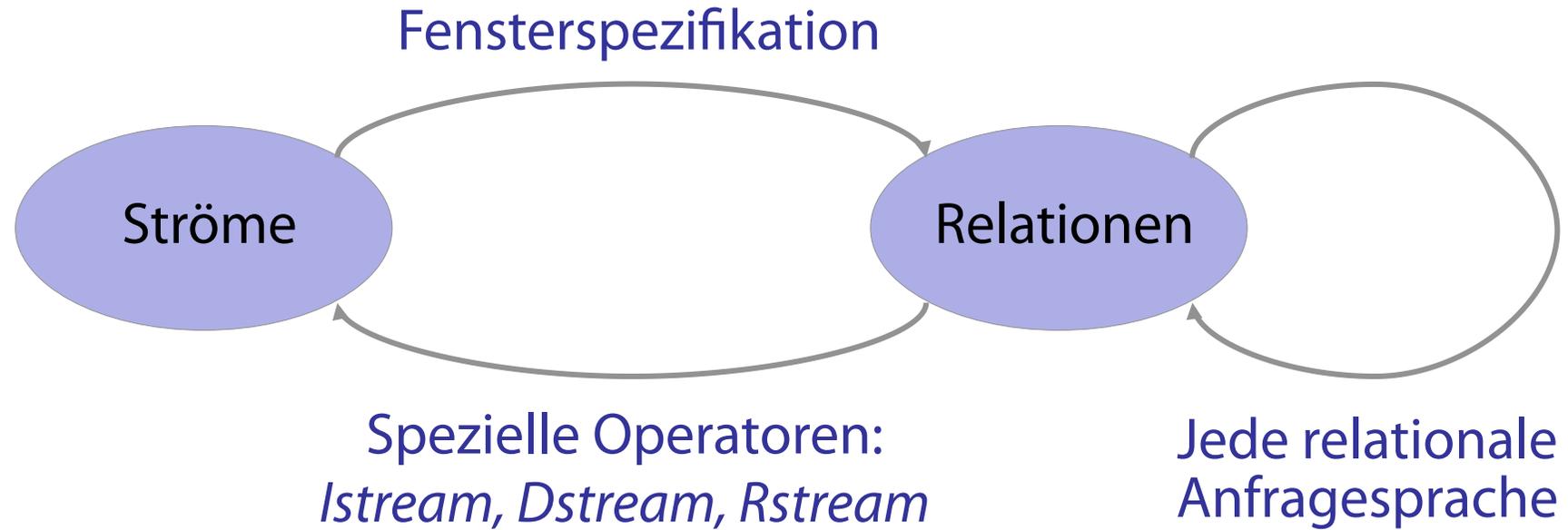
Überblick

- Formale Definition für Relationen und Ströme
- Formale Konvertierung zwischen ihnen
- Abstrakte Semantik
- Konkrete Sprache: CQL
- Syntaktische Besonderheiten
- Äquivalenztransformationen zwischen Anfragen
- Benchmarks

Relationen und Ströme

- Annahme: Globale, diskrete, geordnete Menge von Zeitpunkten
- Relation
 - Bildet *Zeitpunkte T* auf *Tupelmengen R* ab
- Strom
 - Menge von *(Tupel, Zeitstempel)*-Elemente

Konversion



Konversion – Definitionen

- Strom-zu-Relation-Operator $S[\dots]$
 - $S[W]$ 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,
- $Istream(R) := \{ (x, m-n) \mid$
 $(x, m) \in R$ zum Zeitpunkt T ,
 $(x, n) \in R$ zum Zeitpunkt $T-1$,
 $m-n > 0 \}$
 \cup
 $\{ (x, m) \mid$
 $(x, m) \in R$ zum Zeitpunkt T ,
 $(x, n) \notin R$ zum Zeitpunkt $T-1 \}$

CQL: Ansatz

- Man nehme **relationale Anfragesprache SQL** und ergänze **Stromkonzept**
- Man kann Ströme anstelle von Relationen referenzieren
 - Ströme müssen mit Fensterspezifikation zur Relation konvertiert werden (Standardfenster: $[\infty]$)
- Man kann Relationen in Ströme konvertieren
 - Für “ausströmende” Ergebnismengen
 - Für Fenster über Relationen (Konvertierung zurück in eine Relation)

Anfrageergebnis zum Zeitpunkt T

- Verwende all Relationen zum Zeitpunkt T
- Verwende all Ströme bis zum Zeitpunkt T , in eine Relation konvertiert
- Bestimme relationales Ergebnis
- Konvertiere Ergebnis ggf. in einen Strom

Abstrakte Semantik – Beispiel 1

```
Select F.clerk, Max(O.cost)
From O [ $\infty$ ], 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 [ $\infty$ ], 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 [ $\infty$ ], 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($\langle \text{clerk}, \text{max} \rangle, T$) wenn $\langle \text{clerk}, \text{max} \rangle$ 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

Einfachstes Szenario

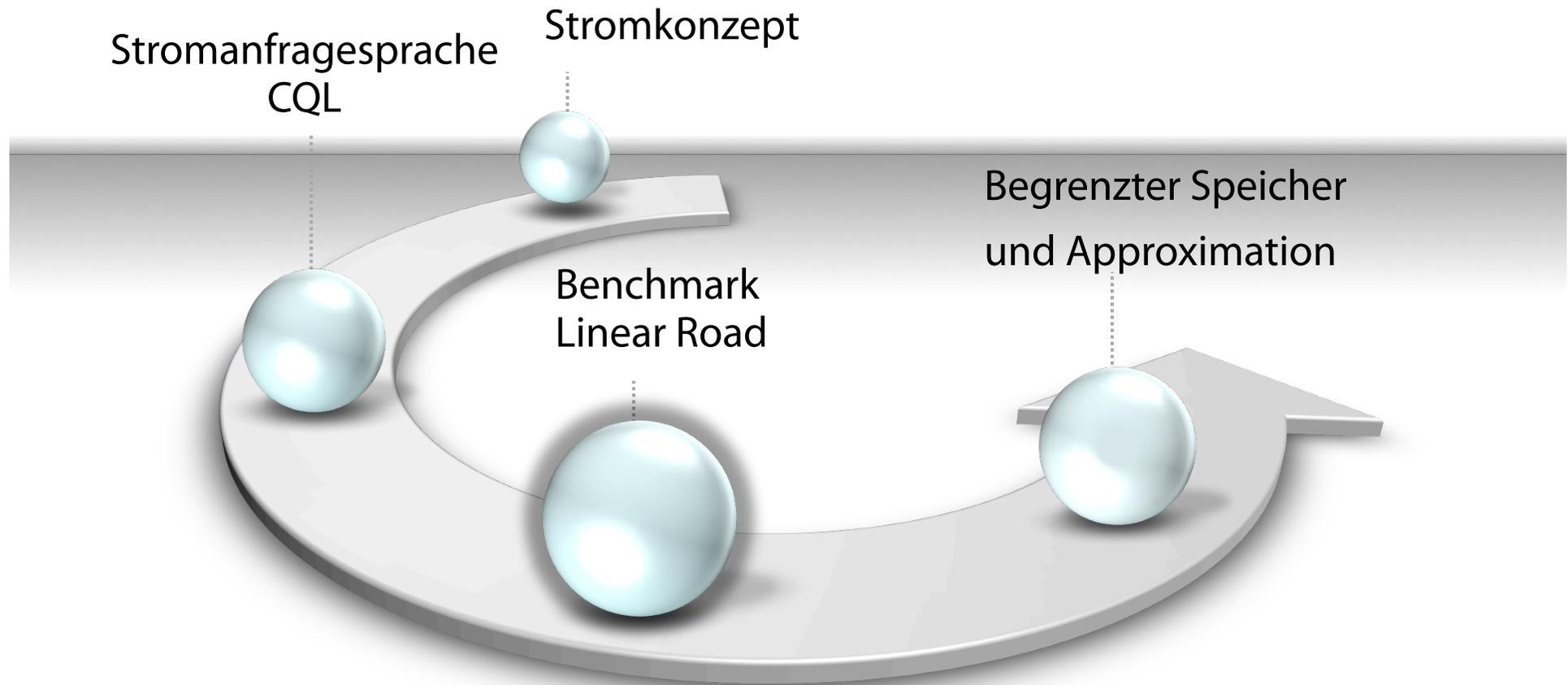
- Globale Systemuhr zur automatischen Assoziation von Zeitstempeln bei Eintritt von Tupeln in das DSMS (Systemzeit)
- Ergebnistupel basieren auf Systemzeit

Komplexeres Szenario

- Zeitstempel repräsentieren Anwendungszeit, außerhalb des DSMS vergeben
 - Tupel können außerhalb der Reihenfolge im DSMS ankommen
 - Eventuell kommen viele „neuere“ Tupel bevor Tupel mit alten Zeitstempeln kommen
- Anfrageergebnisse in Anwendungszeit
- In CQL nicht von Sprachsemantik erfasst

Non-Standard-Datenbanken

Von temporalen Datenbanken zu Stromdatenbanken



Konkrete Sprache – CQL

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

Die einfachste CQL-Anfrage

Select * From Strm

Erzeugt einen Strom als Kopie von *Strm*

- Standardfenster $[\infty]$ für *Strm*
- Standardergebnis *Istream*

Einfache Verbundanfrage

Select * From Strm, Rel

Where Strm.A = Rel.B

- Standardfenster $[\infty]$ 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 $[\infty]$ 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

Äquivalenztransformationen

- Relationale Äquivalenzen sind direkt auf relationale Konstrukte anwendbar
 - Vorteil der Integration von SQL
 - Optimierungstechniken übertragbar
- Weitere Transformation für Stromkonstrukte
 - Fensterreduktion
 - Kommutativität von Filterfenstern

Fensterreduktion: Aufgabe

Select Istream(L) From S [∞] Where C

ist äquivalent zu

Select Rstream(L) from S [Now] Where C

Warum Rstream und nicht Istream in der zweiten Anfrage?

Fensterreduktion

Select Istream(L) From S [∞] Where C

ist äquivalent zu

Select Rstream(L) from S [Now] Where C

- Betrachte Strom $\langle 5 \rangle, \langle 5 \rangle, \langle 5 \rangle, \langle 5 \rangle, \dots$
- Die erste Anfrage kommt durch Standardannahmen häufig vor
- In einer einfachen Implementierung ist die zweite Form weitaus effizienter

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**]

Einige Anmerkungen

- *Dstream* in der Praxis selten verwendet, aber notwendig
- Partitionierende zeit-basierte Fenster nicht benötigt
- Tupelbasierte Fenster nichtdeterministisch bei mehrfach vorkommenden Zeitstempeln
- Äquivalenz von Anfragen gegenüber SQL erheblich schwieriger zu verstehen

Kommutativität von Filterfenstern:

Wann ist

Select L From S [window] Where C

äquivalent zu

Select L

From (Select L From S Where C) [window]

Ist diese Transformation immer vorteilhaft?

Einschränkungs-basierte Transformationen

- Siehe die Beispielanfrage)

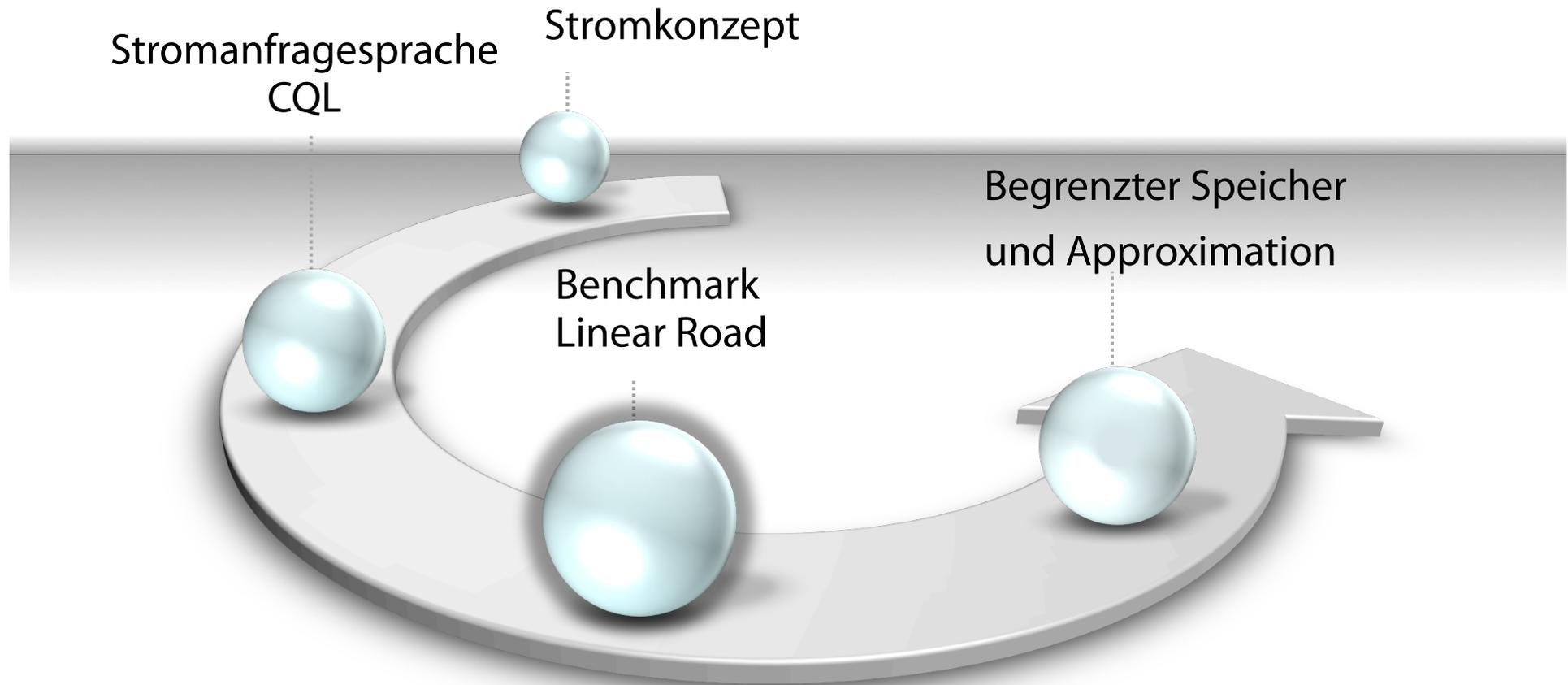
```
Select Sum(O.cost)
From Orders O, Fulfillments F [Range 1 Day]
Where O.orderID = F.orderID
```

- Falls Bestellungen immer in einer Woche bearbeitet:

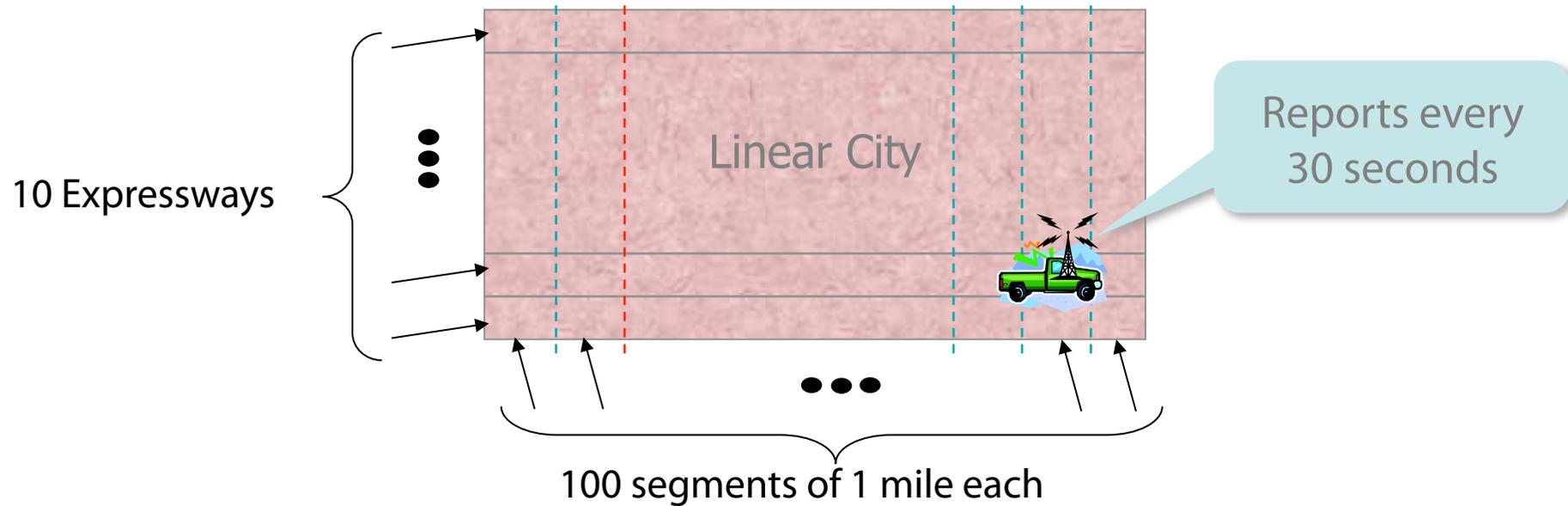
```
Select Sum(O.cost)
From Orders O [Range 8 Days],
      Fulfillments F [Range 1 Day]
Where O.orderID = F.orderID
```

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
```

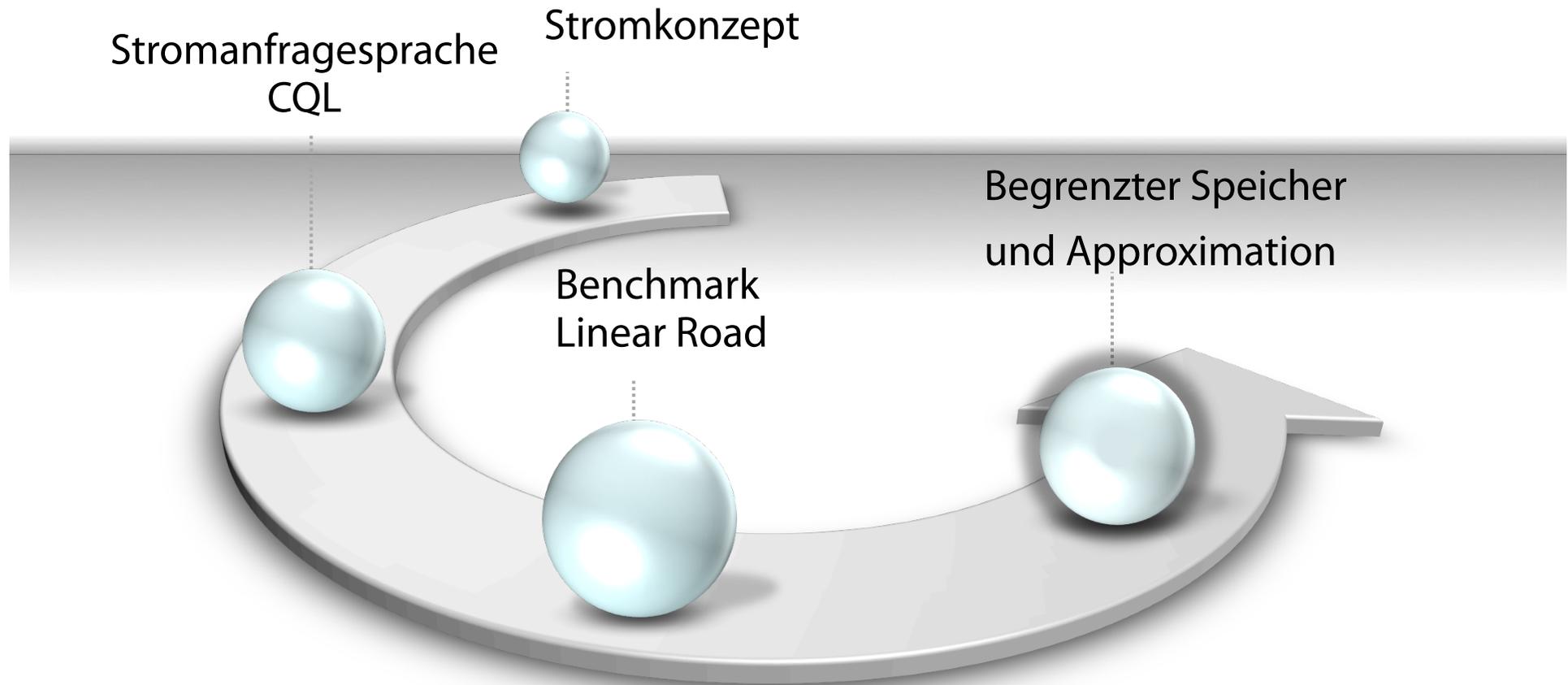
SegVolume:

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



Non-Standard-Datenbanken

Von temporalen Datenbanken zu Stromdatenbanken



Stromverarbeitungsmodelle – interner Speicher A[·]

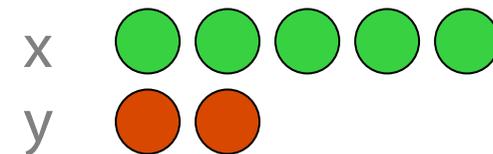
- **Absolute Werte (Zeitreihenmodell)**

- Jeder neue Wert pro Zeiteinheit x wird gespeichert
(x aus Menge von Zeitpunkten)

- **Akkumulierte Werte**

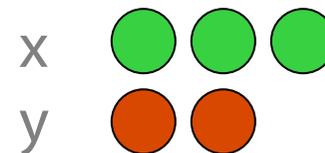
- **Nur Eingänge (Akkumulation von Zählern)**

- Zähler immer > 0
 - Multimengenmodell
 - (x, y sind Objekte bzw. Objekttypen)



- **Eingänge und Abgänge (Drehkreuzmodell)**

- Multimengenmodell
 - (x, y sind Objekte bzw. Objekttypen)



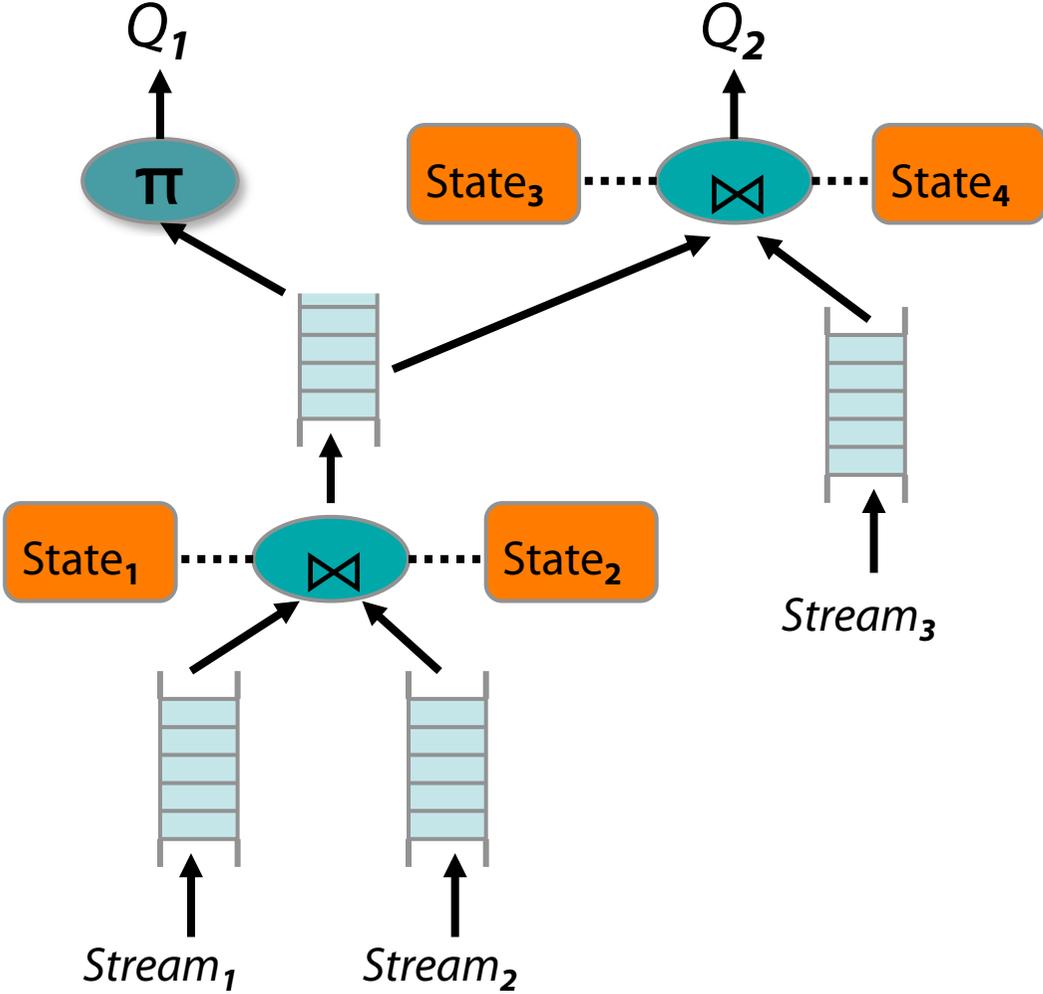
Implementierung eines DSMS – Designeinflüsse

- **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, allmähliche Approximation von korrekten Ausgaben (graceful approximation)

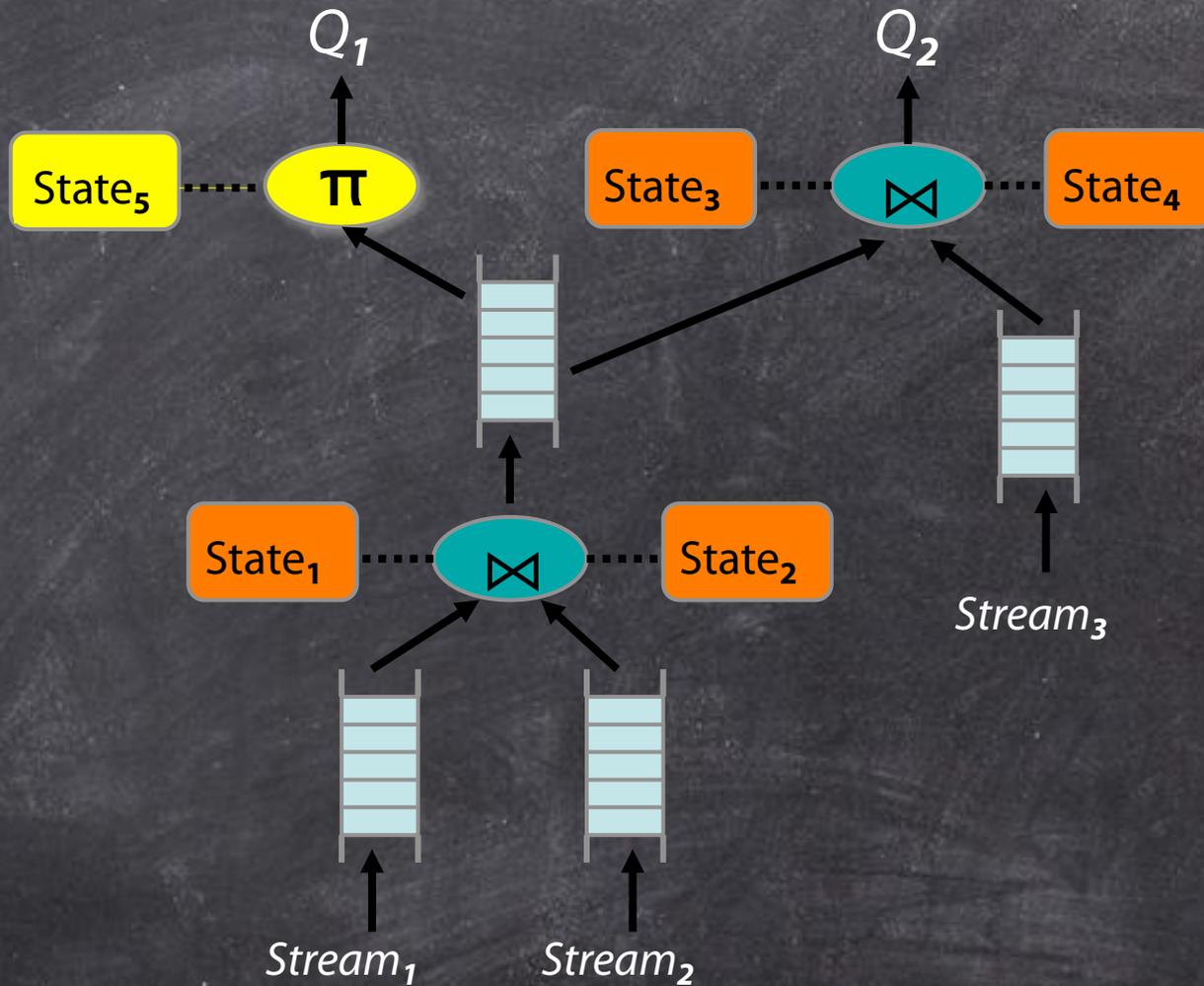
Anfrageverarbeitung

- 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

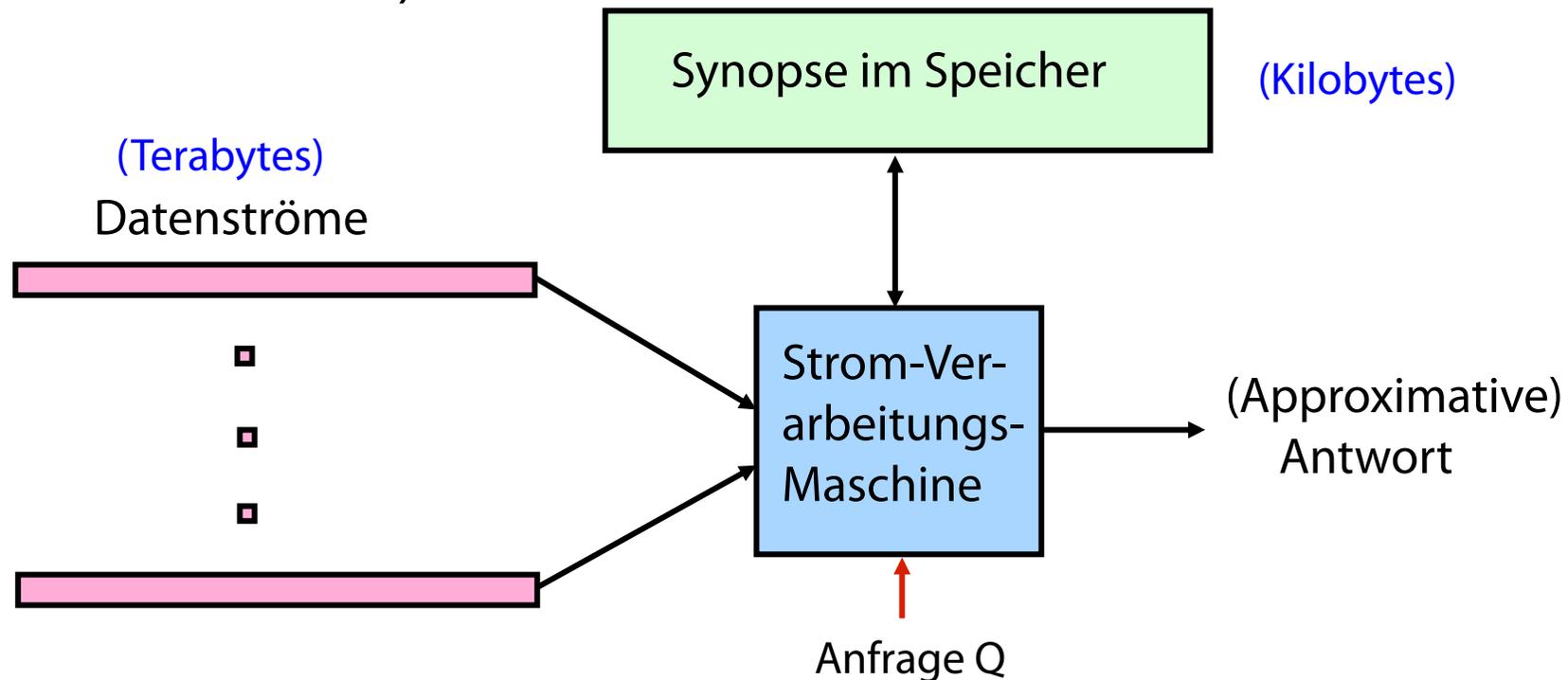
- Fenster sind Möglichkeiten, den Nutzer selbst kleine Einheiten zu definieren zu lassen
- Nicht immer kann man große Fenster (vgl. $S[\infty]$) wegoptimieren
- Auch bei "kleinem" Fenster kann bei einem "Eingabestoß" (burst) eine große Relation entstehen
- → Problematische Verzögerungen
- Auch bei blockierenden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

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 $\pm 10\%$ des korrekten Ergebnisses
 - **Genereller:** finde $(1 \pm \epsilon)$ -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

Approximation

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

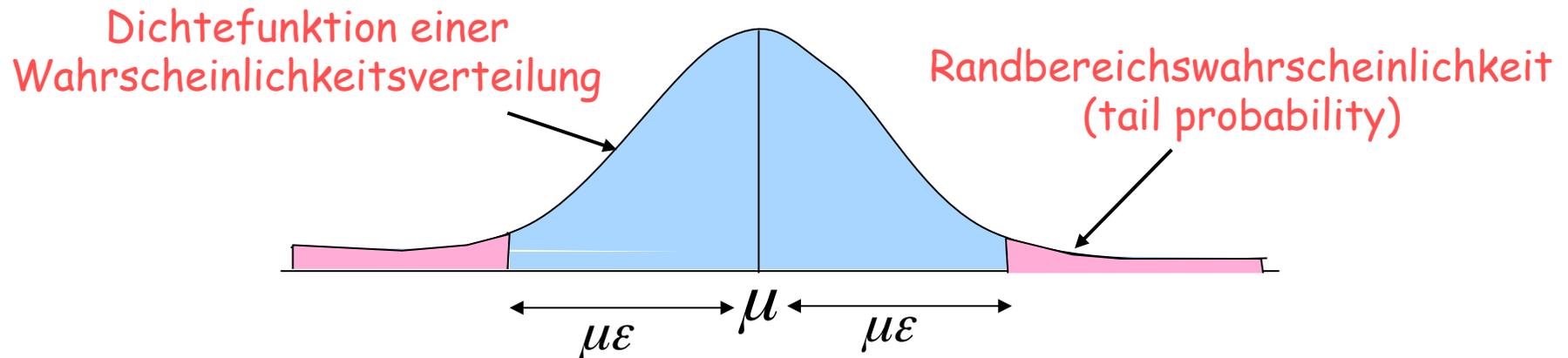


Probabilistische Garantien

- Benutzerbestimmte (ε, δ) -Approximationen
 - Beispiel: Tatsächliche Antworten innerhalb von Faktor 1.1 with 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)



- Basisungleichungen: Sei X eine Zufallsvariable mit Erwartungswert μ und Varianz $\text{Var}[X]$. Dann gilt für alle $\epsilon > 0$

Markov:

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

Chebyshev:

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

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 \rightarrow Mittel der ungeraden Elemente in S Antwort: 5
- agg = count \rightarrow Mittel über Summanden abgel. aus e in S mit
 - n falls e ungerade Antwort: $12 \cdot 3/4 = 9$
 - 0 falls e gerade

Randbereichsschätzungen für Summen

- **Hoeffding-Ungleichung:** Seien X_1, \dots, X_m unabhängige Zufallsvariablen mit $0 \leq X_i \leq r$.

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

Dann gilt für alle $\varepsilon > 0$

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

- 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 Zählfragen:
 - m ist Größe der Stichprobe S (4 im Beispiel)
 - r ist Anzahl der Elemente n im Strom (12 im Beispiel)

Randbereichsschätzungen für Summen (2)

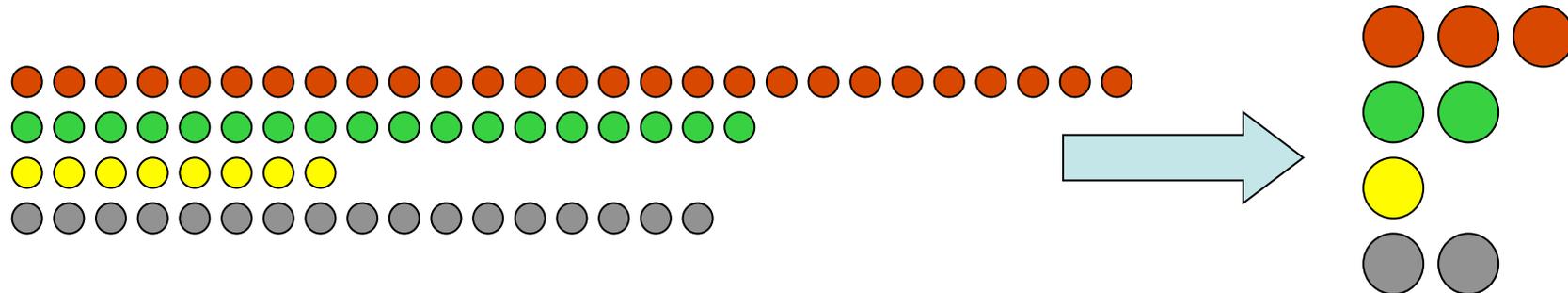
Für Bernoulli-Versuche (zwei mögliche Erg.) sogar starke Eingrenzung möglich:

- Chernoff Abschätzung: Sei X_1, \dots, 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 $\varepsilon > 0$

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

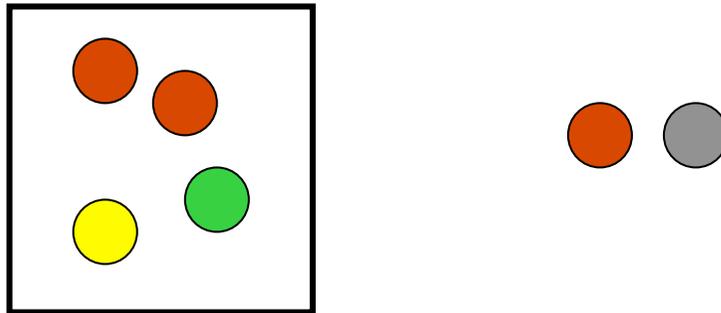
- Verwendung für Anzahlanfrage (count queries):
 - m ist Größe der Stichprobe S (4 im Beispiel)
 - p ist Anteil der ungeraden Elemente im Strom (2/3 im Beispiel)
- Anmerkung: Chernoff-Abschätzung liefert engere Grenzen für Anzahlanfragen als die Hoeffding-Ungleichung

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



- Übernahme 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)

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”

$$\begin{aligned} & \frac{1}{i} \times \left(1 - \frac{1}{i+1}\right) \times \left(1 - \frac{1}{i+2}\right) \dots \left(1 - \frac{1}{n-1}\right) \times \left(1 - \frac{1}{n}\right) \\ &= \frac{1}{\cancel{i}} \times \frac{\cancel{i}}{\cancel{i+1}} \times \frac{\cancel{i+1}}{\cancel{i+2}} \dots \frac{\cancel{n-2}}{\cancel{n-1}} \times \frac{\cancel{n-1}}{n} = 1/n \end{aligned}$$

- 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

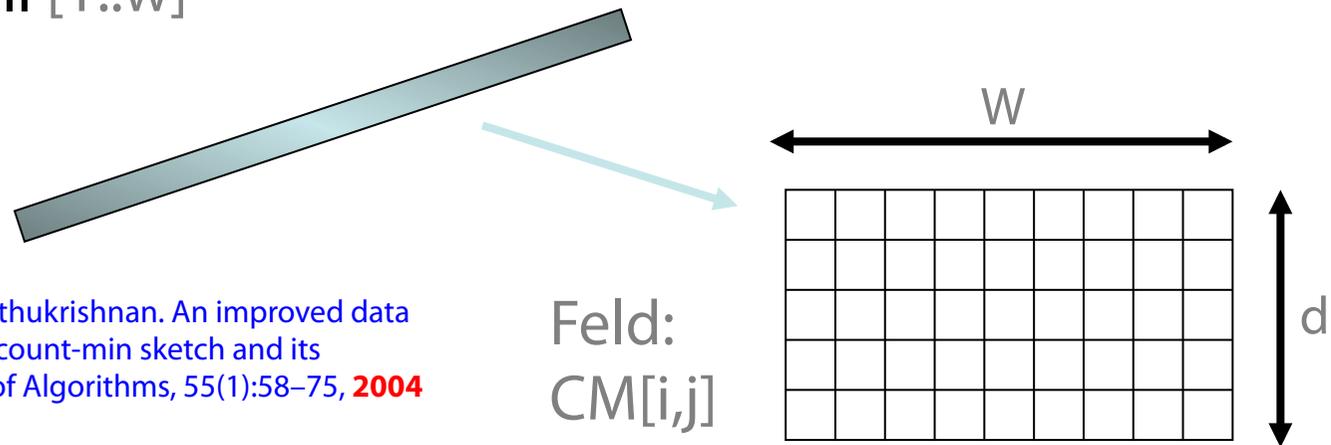


- Jedes Element hat gleiche Chance, kleinstes Element zu werden, also gleichverteilt
- Anwendung auf mehrere Ströme, dann Zusammenführung

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

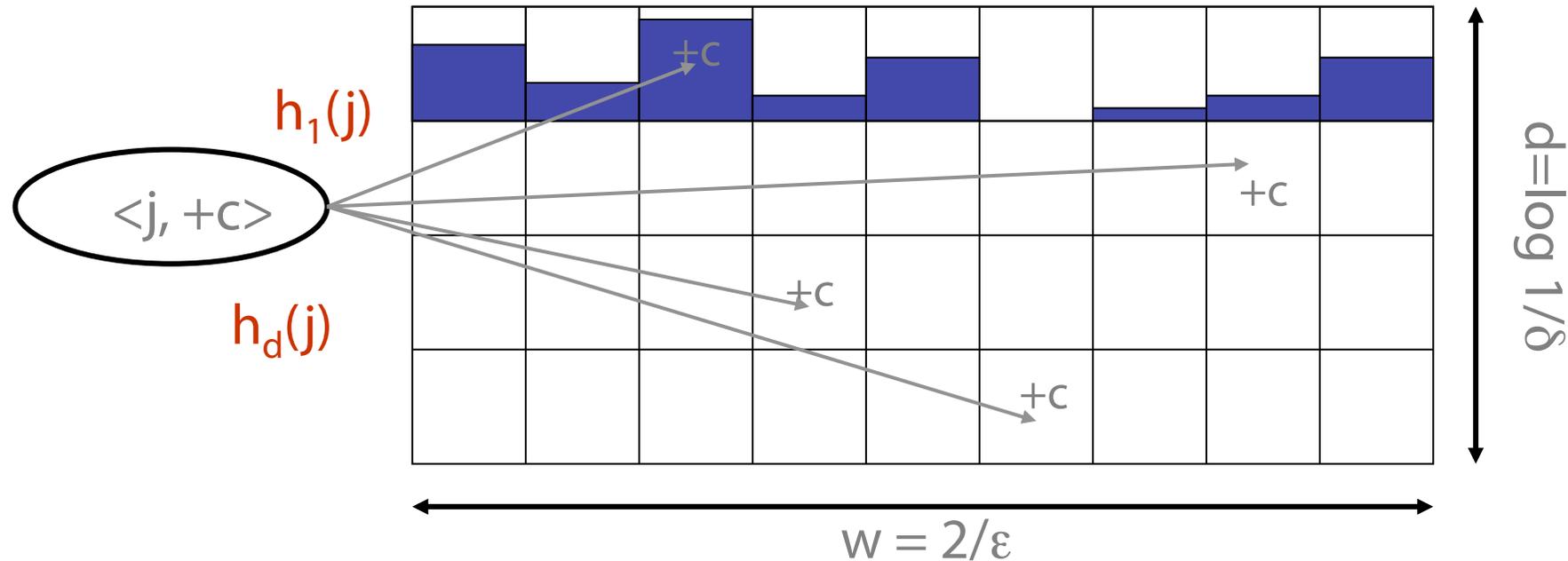
Count-Min-Skizzen (CM-Skizzen)

- Einfache Skizzenidee, als Basis für viele Strom-Untersuchungsaufgaben (stream mining tasks)
 - hauptsächlich für “Anzahl Elemente pro Typ” (item frequenciens)
 - 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
- Parallelisierung: Verschmelzung eintragsweise durch Summierung verschiedener CMs möglich
- Schätze $A[j]$ über den Ausdruck $\min_k \{ CM[k, h_k(j)] \}$

CM-Skizzen – Garantien

- CM-Approximierungsfehler bei Punktanfragen kleiner als $\epsilon \|A\|_1$ mit Platzbedarf $O(1/\epsilon \log 1/\delta)$ [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 begrenztbar 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

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

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

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

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

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

$$P[X_{k,j} \geq \varepsilon \|A\|_1] = P[X_{k,j} \geq \underbrace{2E[X_{k,j}]}_{(1+\varepsilon)\mu}] \leq \frac{1}{2}$$

Markov-
Abschätzung \rightarrow

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

s. Folie 66

$$P[\forall k. X_{k,j} \geq \varepsilon \|A\|_1] \leq \underbrace{\frac{1}{2} \cdot \frac{1}{2} \cdot \dots \cdot \frac{1}{2}}_d = \frac{1}{2^d} = \frac{1}{2^{\log_2 \frac{1}{\delta}}} = \delta$$

CM-Skizzen – Analyse

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

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

Anzahl **verschiedener** Werte abschätzen

- Aufgabe: Finde Anzahl der verschiedenen Werte in einem Stream von Werten aus $[1, \dots, N]$ (**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:

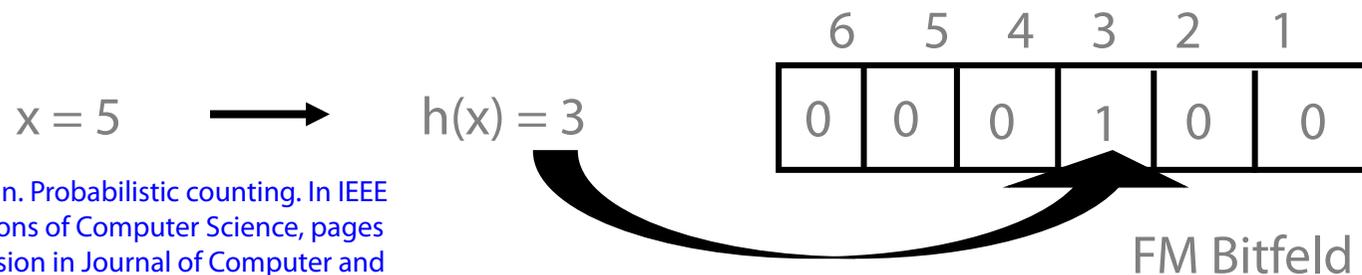
3 2 5 3 2 1 7 5 1 2 3 7

Anzahl der verschiedene Werte: 5

- Schwierig mit CM (gedacht für Multimengen)

FM-Skizzen [Flajolet, Martin' 85]

- Verwende Hashfunktion zur Abbildung von Eingabeelementen auf i mit Wahrscheinlichkeit 2^{-i}
 - also $\Pr[h(x) = 1] = 1/2$, $\Pr[h(x) = 2] = 1/4$, $\Pr[h(x)=3] = 1/8 \dots$
 - Konstruiere $h()$ aus gleichverteilter Hashfunktion durch Zählen der Null am Ende
- Aufbau FM-Skizze= Bitfeld von $L = \log N$ Bits
 - Initialisiere Bitfelder of 0
 - Für jeden neuen Wert x , setze $FM[h(x)] = 1$

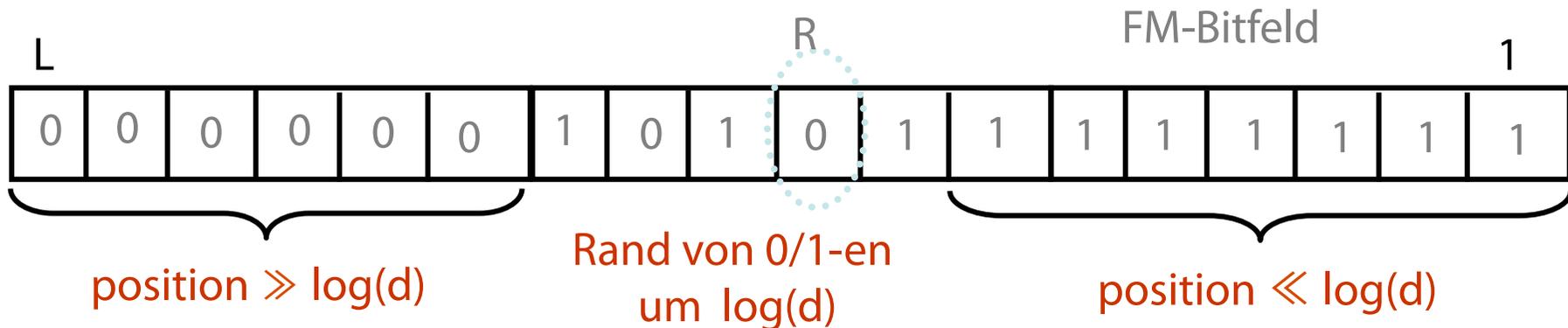


P. Flajolet and G. N. Martin. Probabilistic counting. In IEEE Conference on Foundations of Computer Science, pages 76–82, 1983. Journal version in Journal of Computer and System Sciences, 31:182–209, 1985.



FM-Skizzen – Analyse

- Bei d verschiedenen Werten, erwarte $d/2$ in FM[1], $d/4$ in 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

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

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

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

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

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

$$d \approx \frac{2^R}{\Phi} \approx c 2^R \quad c \approx 1.3..$$

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 \approx 30% Fehler, 100 Funkt. $<$ 10% Fehler
- *Bei Löschung:* Verwende Zähler statt Bits
 - +1 für Einfügungen, -1 für Löschungen
- *Komposition:* komponentenweise OR bzw. +

$$\begin{array}{cccccc} 6 & 5 & 4 & 3 & 2 & 1 \\ \boxed{0} & \boxed{0} & \boxed{1} & \boxed{0} & \boxed{1} & \boxed{1} \end{array} + \begin{array}{cccccc} 6 & 5 & 4 & 3 & 2 & 1 \\ \boxed{0} & \boxed{1} & \boxed{1} & \boxed{0} & \boxed{0} & \boxed{1} \end{array} = \begin{array}{cccccc} 6 & 5 & 4 & 3 & 2 & 1 \\ \boxed{0} & \boxed{1} & \boxed{1} & \boxed{0} & \boxed{1} & \boxed{1} \end{array}$$

- Schätze $|S_1 \cup \dots \cup S_k| = \text{Kardinalität der Vereinigungsmenge}$

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 ...)

Praktische Betrachtungen

- Algorithmen sind einfach und schnell
 - Skizzen geeignet für Millionen von Operationen pro Sekunden auf Standard-Hardware
 - Begrenzung durch I/O-Aspekte
- Implementiert in praktischen Systemen:
 - AT&Ts Gigascope-System oder Sprints CMON-System für Online-Netzwerk-Analyse
 - Googles Loganalyse
- Anwendbarkeit für Big Data offensichtlich
- Beispielimplementierungen
 - <http://www.cs.rutgers.edu/~muthu/massdal-code-index.html>

Zusammenfassung

Von Stromdatenbanken zu Big Data

