Non-Standard-Datenbanken

Stromdatenbanken

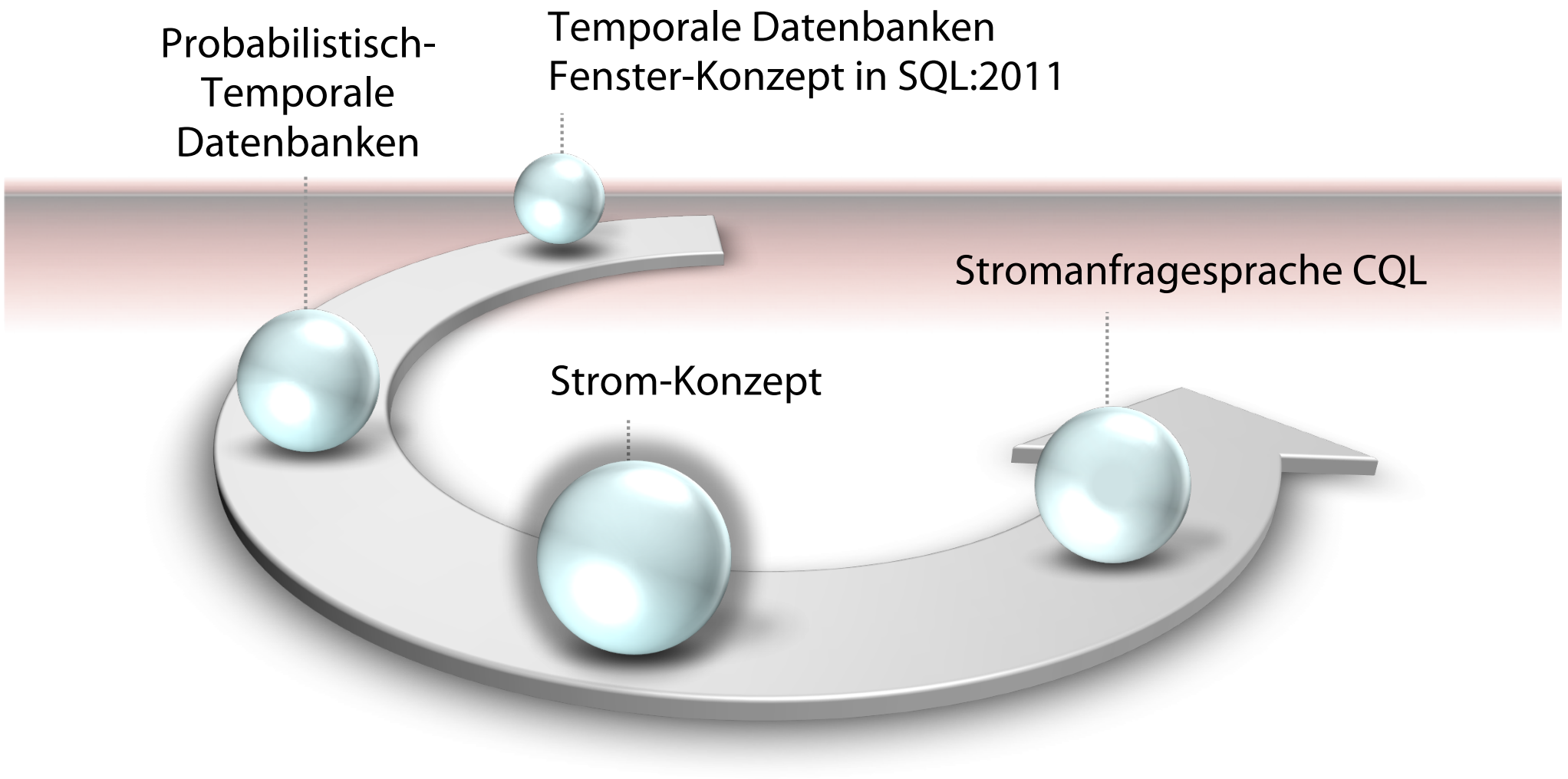
Prof. Dr. Ralf Möller

Universität zu Lübeck

Institut für Informationssysteme

Non-Standard-Datenbanken

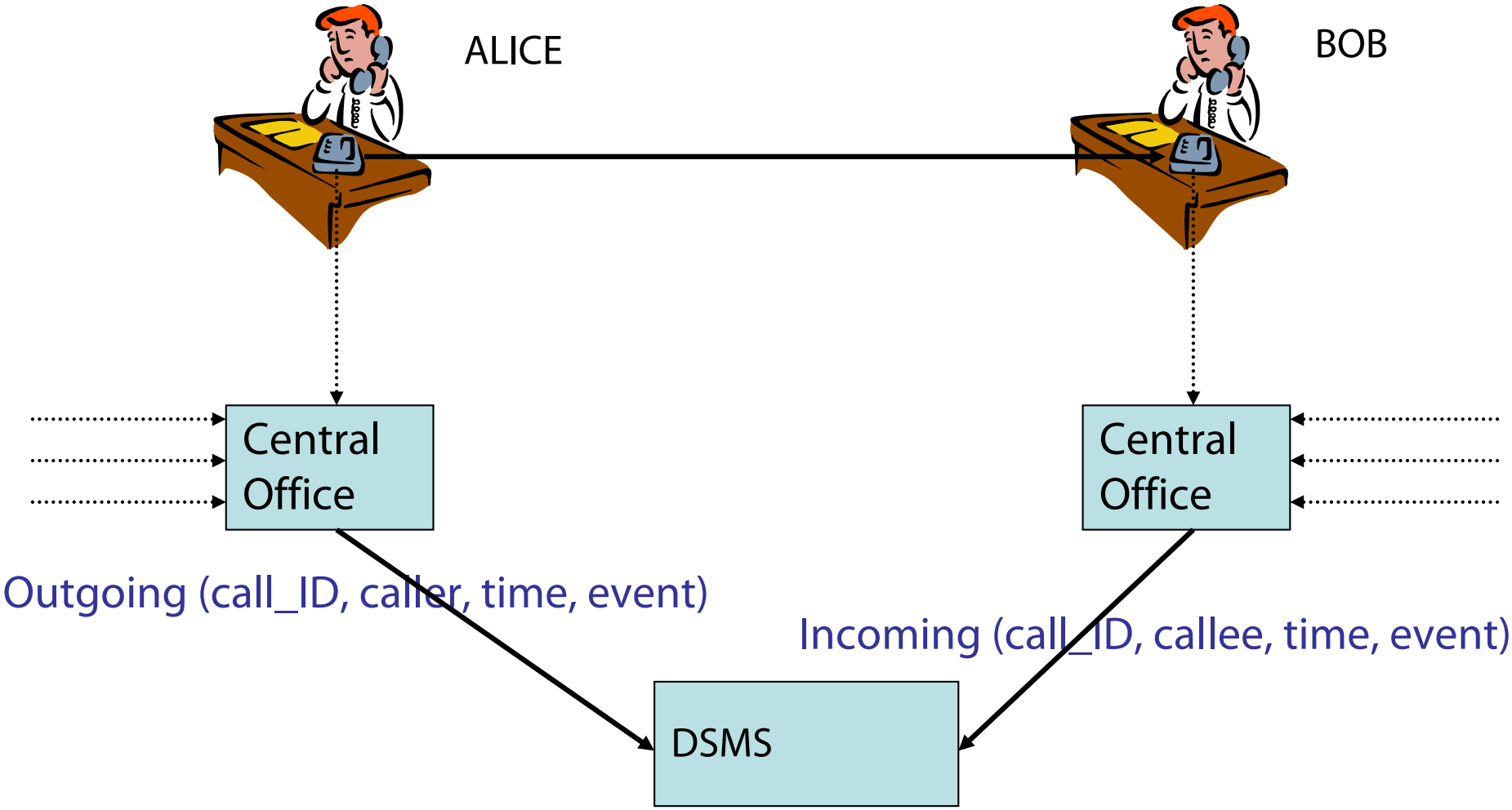
Von temporalen Datenbanken zu Stromdatenbanken



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)

Beispiel 1: Telefondatenauswertung



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

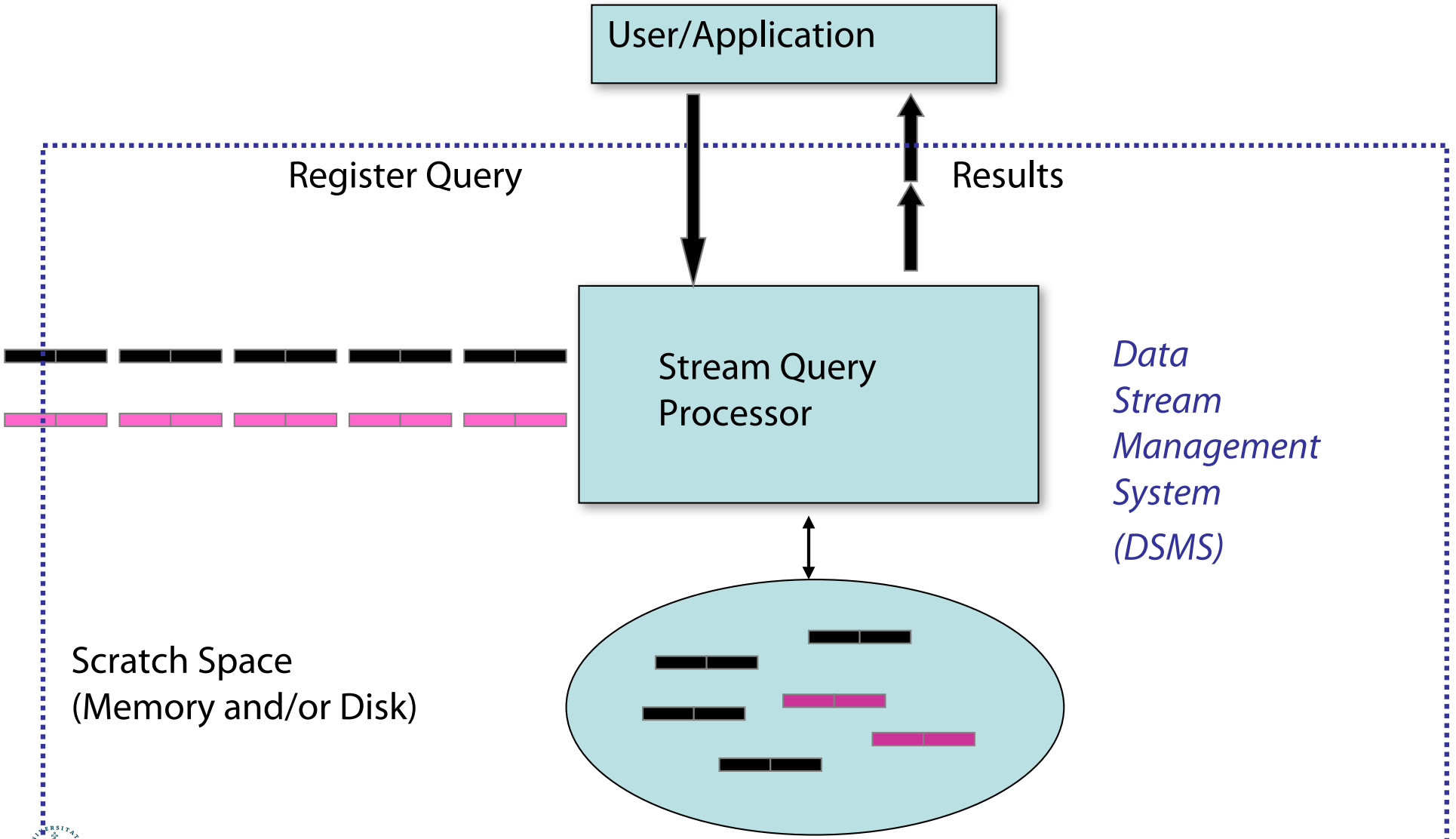
```
HighwayStream( lane, speed, length, timestamp )
```

  - 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

Data Stream Management System



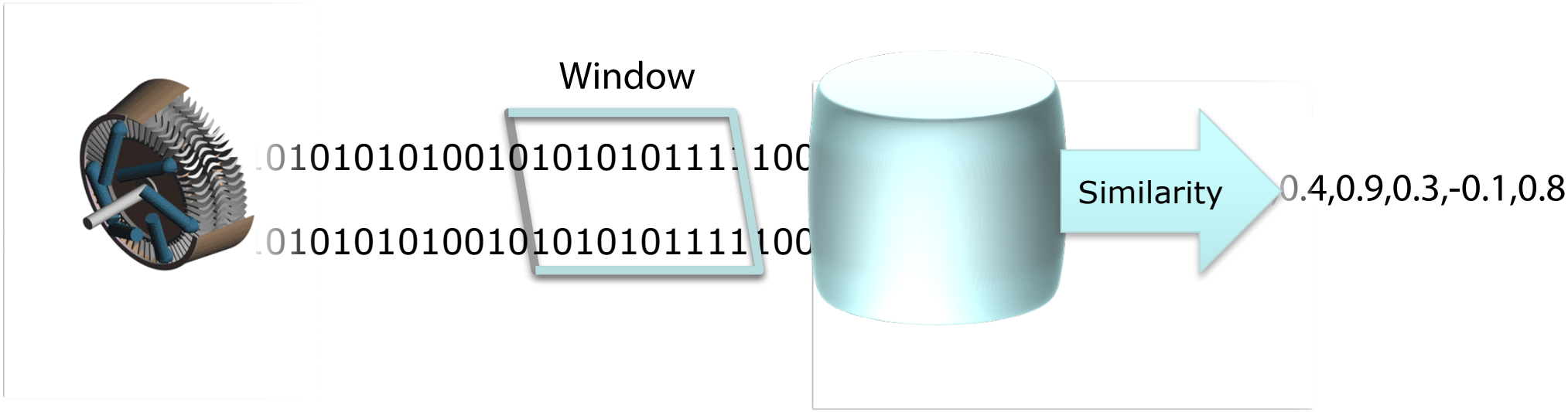
Beispiel: Sensordatenauswertung

- Problem

  Finde Ähnlichkeiten zwischen Messungen zweier Temperatursensoren

- Lösung

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



Fensterkonzept

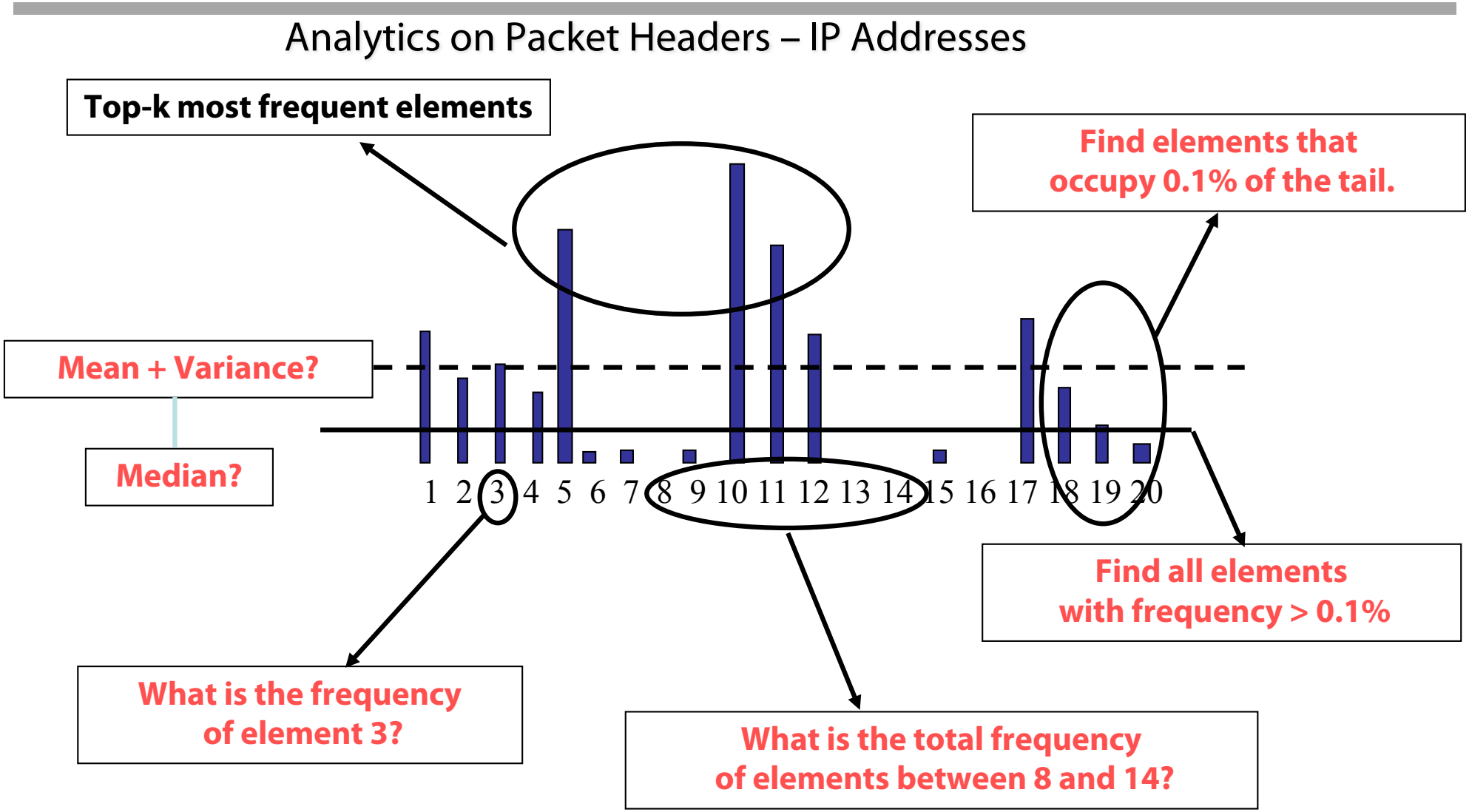
Kein Einfluss von “alten” Daten auf Ergebnis

➔ 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 Zeitpunktfenster [t, t] oder Intervallfenster [0, t]
    - [t – w, t] Anfang und Ende der Gültigkeit (Intervallfenster)

Arten von Anfragen



Rajeev Motwani 2003

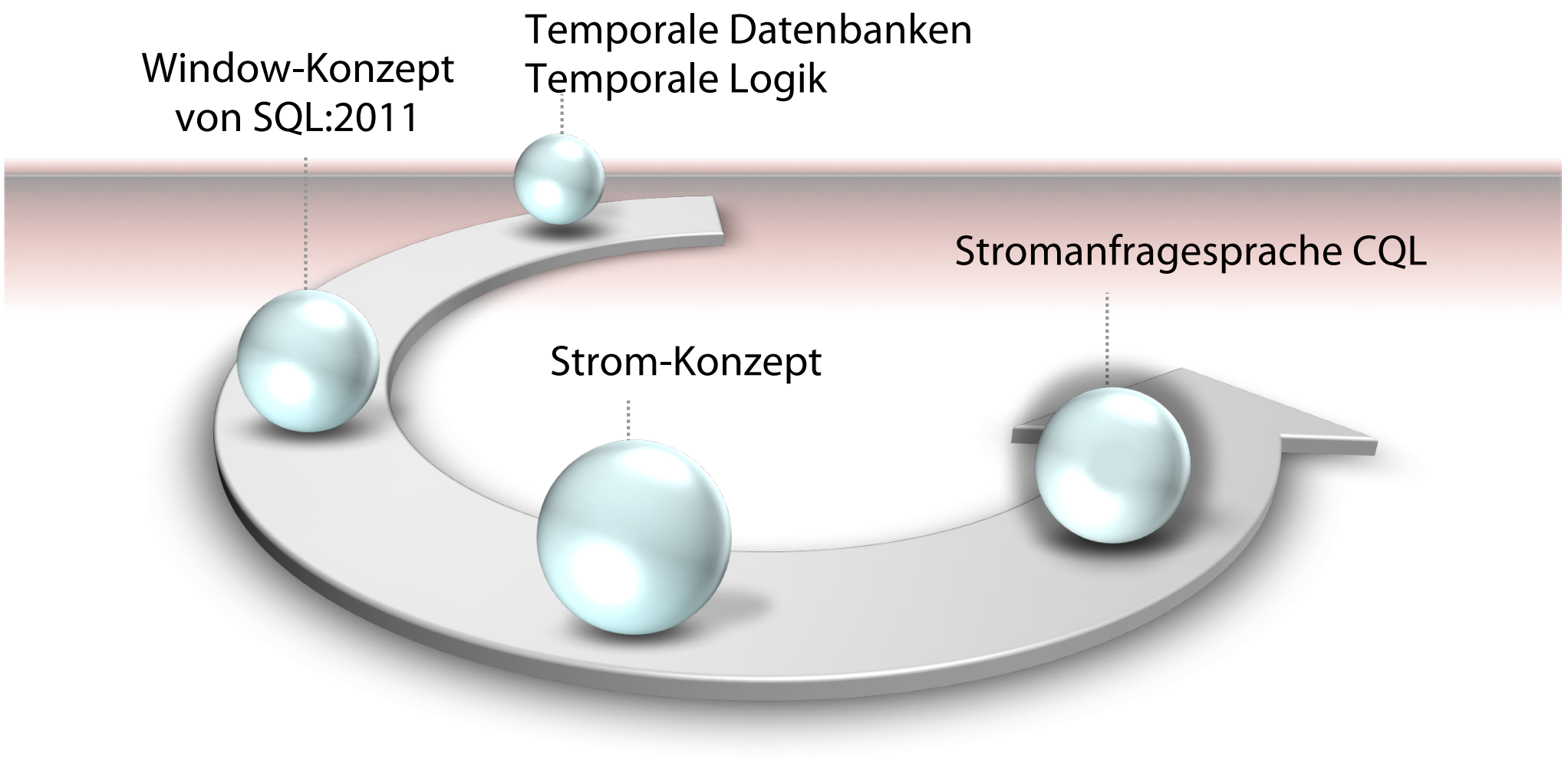
How many elements have non-zero frequency?

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

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, **2006**

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

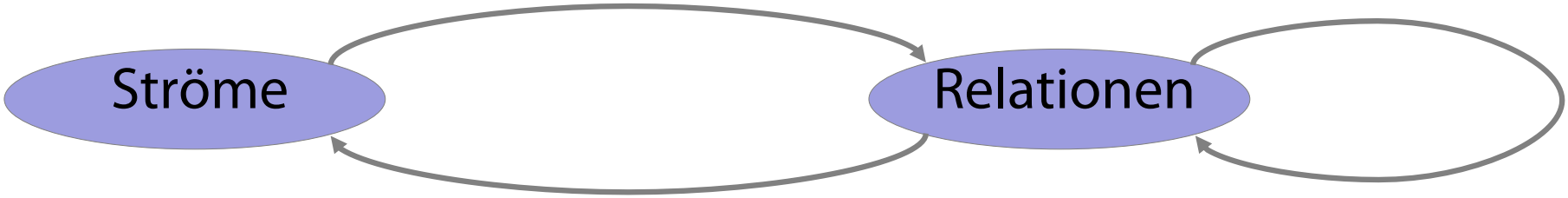
```
Select F.clerk, Max(O.cost)
From Orders O[∞] ,
     Fulfillments F[Partition By clerk Rows 5] 10% Sample
Where O.orderID = F.orderID
Group By F.clerk
```

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 …
  - s: select ... zur Abkürzung
- Anfragen werden beim DSMS registriert (“kontinuierliche” Anfragen)

Konversion

Fensterspezifikation



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 Zeitintervall, 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)* := { $(x, m-n)$ |

  $(x, m) \in R$ zum Zeitpunkt T,

  $(x, n) \in R$ zum Zeitpunkt $T-1$,

  $m-n > 0$ }

  $\cup$

  { (x, m) |

  $(x, m) \in R$ zum Zeitpunkt T,

  $(x, n) \notin R$ zum Zeitpunkt $T-1$ }

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*: Gan… Strom *O* …
  Tupel von *F* als Relation
- Evaluiere Anfrage, ak…
  Zeitpunkt *T*
- **Streamed result:** Neu…
  wenn<clerk,max> sich be…

Was muss die Anfrageoptimierung leisten?

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 zu behandeln, 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

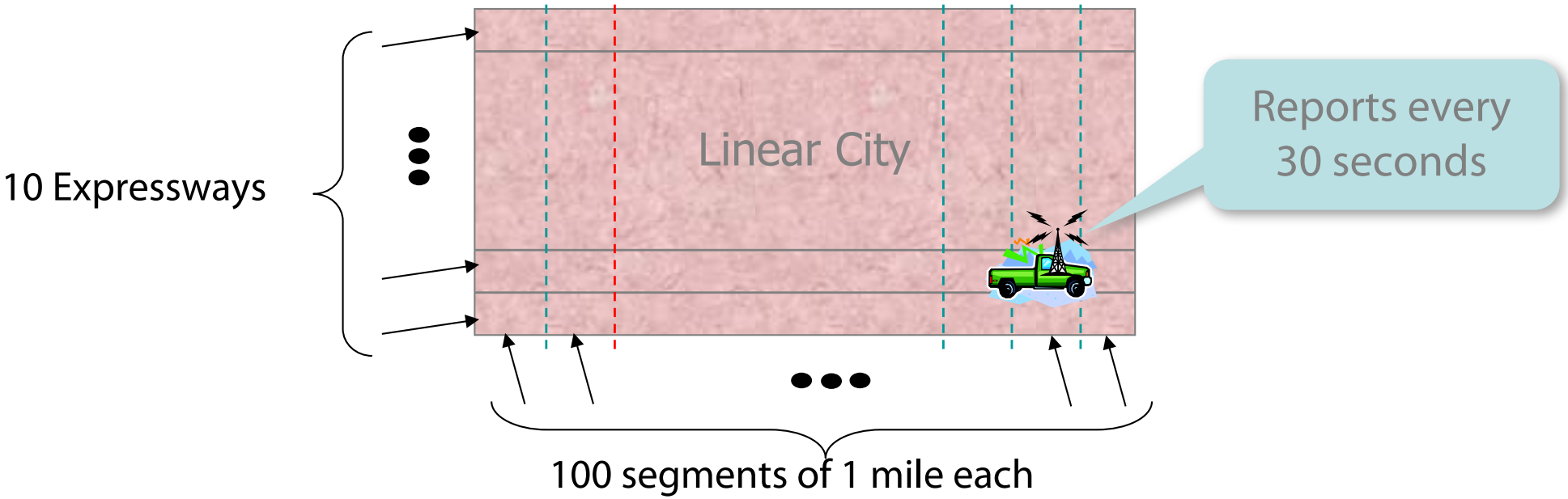
  Select **Rstream**(Avg(A)) From S [Range 5 seconds]

- To emit a result every second

  Select Rstream(Avg(A))

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

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:

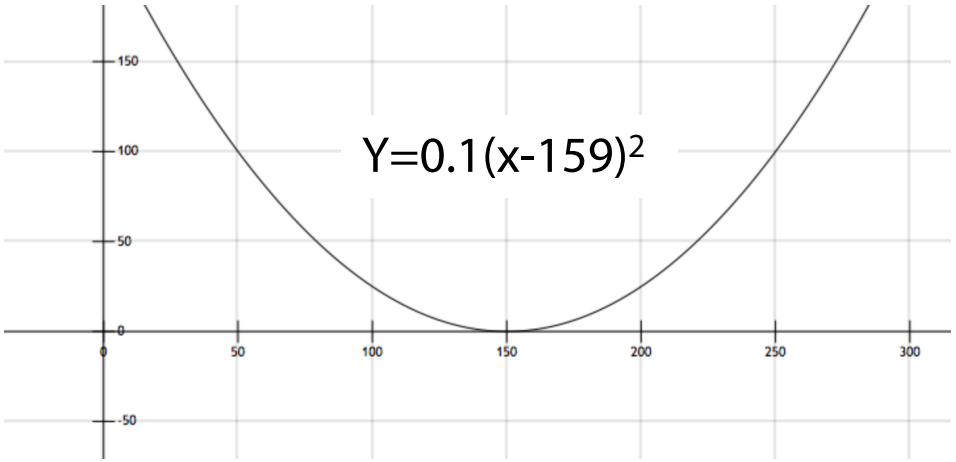
```
Select S.loc, BaseToll * (V.volume – 150)^2
From SegAvgSpeed as S, SegVolume as V
Where S.loc = V.loc and S.avg_speed < 40.0
```

SegAvgSpeed:

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

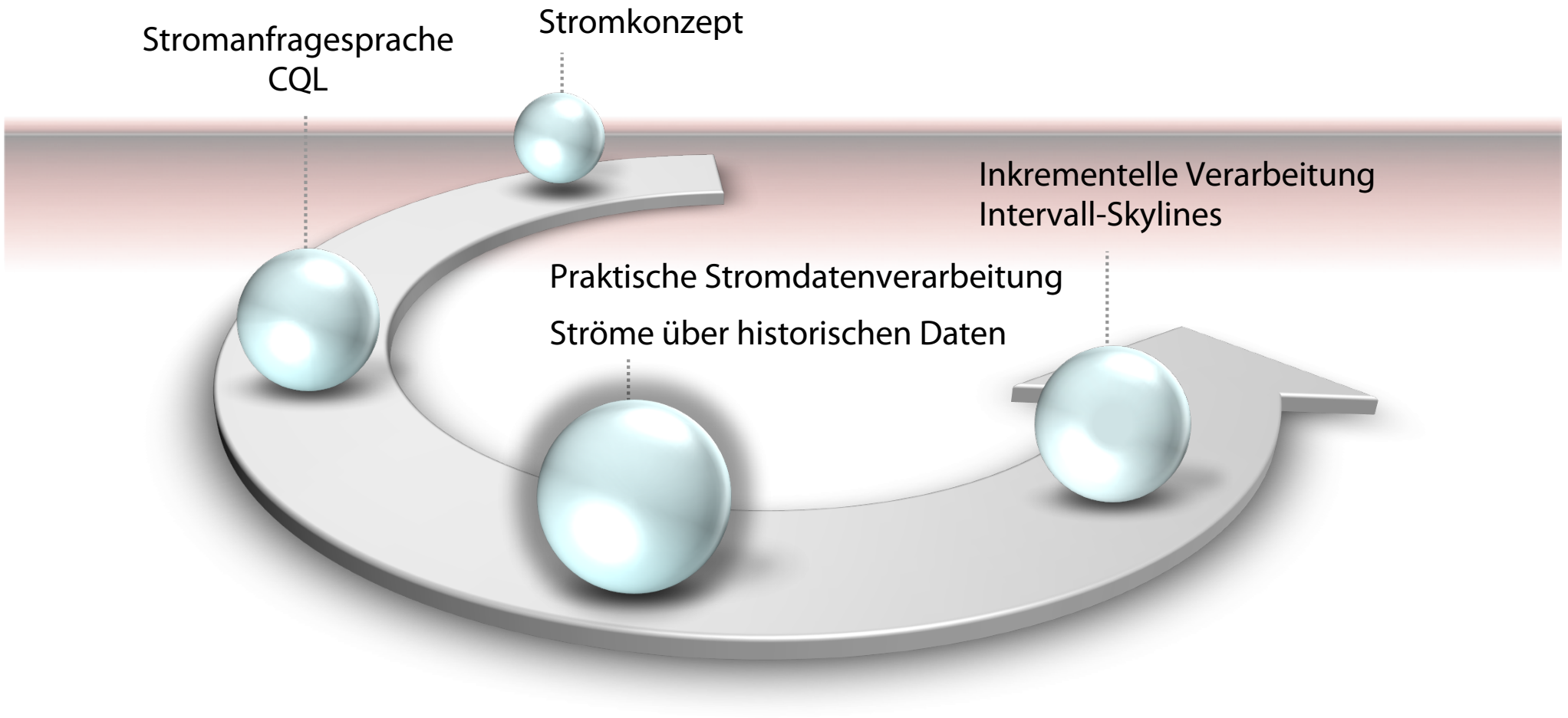
SegVolume:

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



Non-Standard-Datenbanken

Von temporalen Datenbanken zu Stromdatenbanken



Praktische Stromverarbeitung: PipelineDB

- Open-Source, initial basierend auf PostgreSQL 9.5
- Kontinuierliches SQL auf Datenströmen (kontinuierliche Sichten)
  - CREATE STREAM stream (x int, y int, z int);
    INSERT INTO stream (x, y, z) VALUES (0, 1, 2);
  - Fensterkonzept
    CREATE CONTINUOUS VIEW v WITH (max_age = '1 hour') AS
    SELECT COUNT(*) FROM stream
  - Anwendungsspezifische Aggregation
  - Hoher Durchsatz, inkrementell materialisierte Sichten
  - Realzeitanalysen, Überwachung
    - Top-k, Perzentile, Distinct, ….

Weitere Stromdatenverarbeitungsumgebungen

- Apache
  - Flink (flink.apache.org)
  - Kafka (kafka.apache.org)
  - Spark (spark.apache.org)
  - Storm (storm.apache.org)
- Odysseus (odysseus.informatik.uni-oldenburg.de)
- Elastic Stack (www.elastic.co)
- Viele weitere …

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 Approximation von korrekten Ausgaben (graceful approximation)

Fenster-orientierte Verarbeitung

- Fenster sind Möglichkeiten, den Nutzer selbst kleine Einheiten zu definieren zu lassen
- Nicht immer kann man immer größer werdende Fenster (vgl. SQL:2011) 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

QoS-Vereinbarungen

- QoS-Vereinbarung kann beinhalten, dass Eingaben ggf. nicht verarbeitet werden ("Skip"), um nicht zurückzufallen
- Statt Tupel wegzulassen, Approximationstechniken betrachten

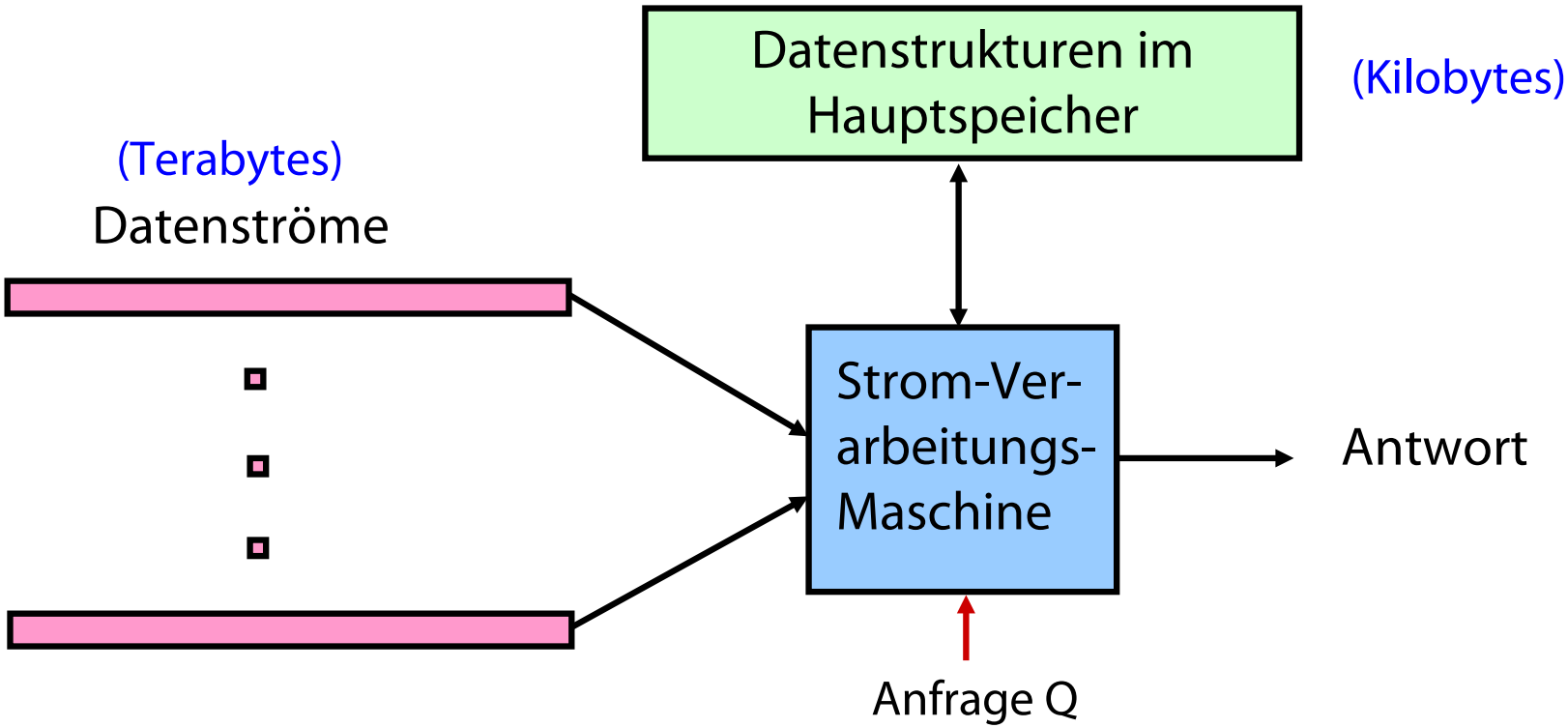
- Aber: Statt Approximation erst einmal inkrementelle Verarbeitung richtig organisieren

Historische Daten

- Fensterorientierte Verarbeitung auch für Anfragen bzgl. historischer Daten
  - Dadurch einfacher Vergleich zwischen alten und aktuellen Daten möglich: Wann gab es eine aktuelle Situation schon einmal?
  - Ausführung von Fenster-basierten Anfragen auf "alten" Daten
  - Approximation auch für Verarbeitung historischer Daten
  - Verarbeitungszeit pro Zeitfenster begrenzbar
- Insbesondere bei überlappenden Fenstern muss eine parallele Verarbeitung nicht notwendigerweise zu Geschwindigkeitsvorteilen führen
  - Prozessoren verarbeiten "gleiche" Daten
  - Daten müssen den Prozessoren zugeführt werden (Aufwand)
  - Lieber Pipeline aufbauen und inkrementelle Verarbeitung mit einem Prozessor

Kontinuierliche und historische Anfragen

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



Beispiel: Elektrizitätsverbrauchsanalyse

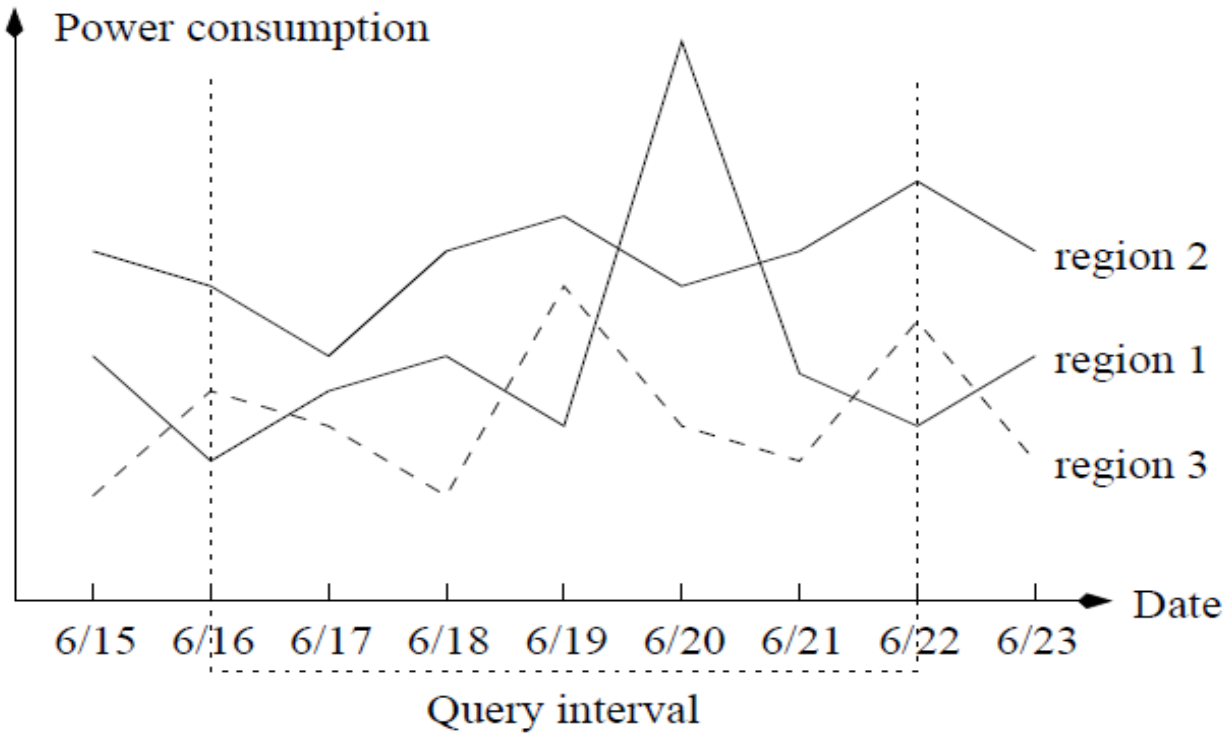


Fig. 1. A set of power consumption time series.

Bin Jiang, Jian Pei, Online Interval Skyline Queries on Time Series,
In Proceedings of the 25th international conference on data engineering (ICDE'09) **2009**

Fensterbasierte Anfragen

Informationsbedarf:

Welche Regionen haben hohen Verbrauch in der Woche vom 16.-22. Juni?

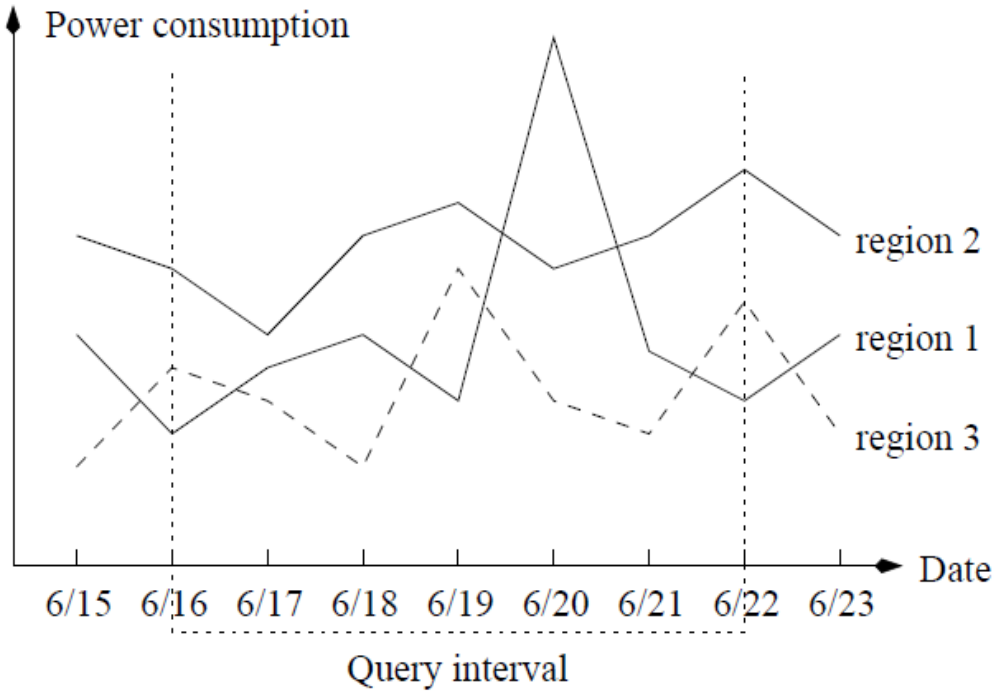


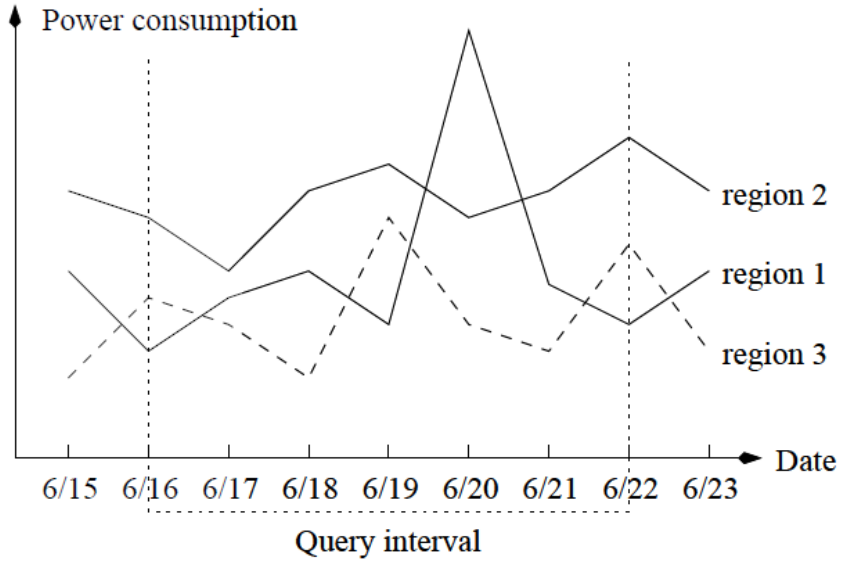
Fig. 1. A set of power consumption time series.

Region 1: **Interessant**
Höchster Tagesverbrauch
am 20. Juni

Region 2: **Interessant**
Höchster Durchschnittsverbrauch
im Anfragefenster

Region 3: **Nicht interessant**
Geringerer Verbrauch als Region 2
an jedem Tag

Intervall-Skyline-Anfragen



Für den Informationsbedarf ist eine Zeitreihe s "interessant", falls im Anfragefenster keine andere Zeitreihe s' liegt, so dass

(1) s' besser ist als s zu mindestens einem Zeitpunkt und

(2) s' nicht schlechter ist als s zu jedem Zeitpunkt

Naiver Ansatz

Zeitreihe: $(t_0,v_0),(t_1,v_1).....(t_n,v_n)$

⇩ Zeitreihe = Punkt im n-dimensionalen Raum

$(v_0,v_1,.....v_n)$

⇩ Skyline-Berechnung

Skyline-Punkte = interessante Zeitreihen

Probleme

(1) Einfache Skyline-Anfragebeantwortungstechniken nicht inkrementell

(2) Insbesondere für hohe Dimensionen (große Fenster) daher nicht effektiv

(3) Für überlappende Zeitfenster im Stromverarbeitungskontext daher nicht geeignet

Intervall-Skyline-Anfrage

- Eine Zeitreihe s dominiert eine Zeitreihe q im Intervall [i:j], geschrieben $s \succ_{[i:j]} q$, falls $\forall k \in [i:j]$ s[k] ≥ q[k] und $\exists l \in [i:j]$: s[l]>q[l]
- Sei S eine Menge von Zeitreihen und [i:j] ein Intervall, dann ist die Intervall-Skyline bzgl. S und [i:j] die Menge der nicht dominierten Zeitreihen aus S in [i:j], geschrieben

$$Sky[i:j] = \{s \in S | \nexists s' \in S, s' \succ_{[i:j]} s\}$$

Bin Jiang, Jian Pei, Online Interval Skyline Queries on Time Series,
In Proceedings of the 25th international conference on data engineering (ICDE'09) **2009**

Energieverbrauchsdiagramm

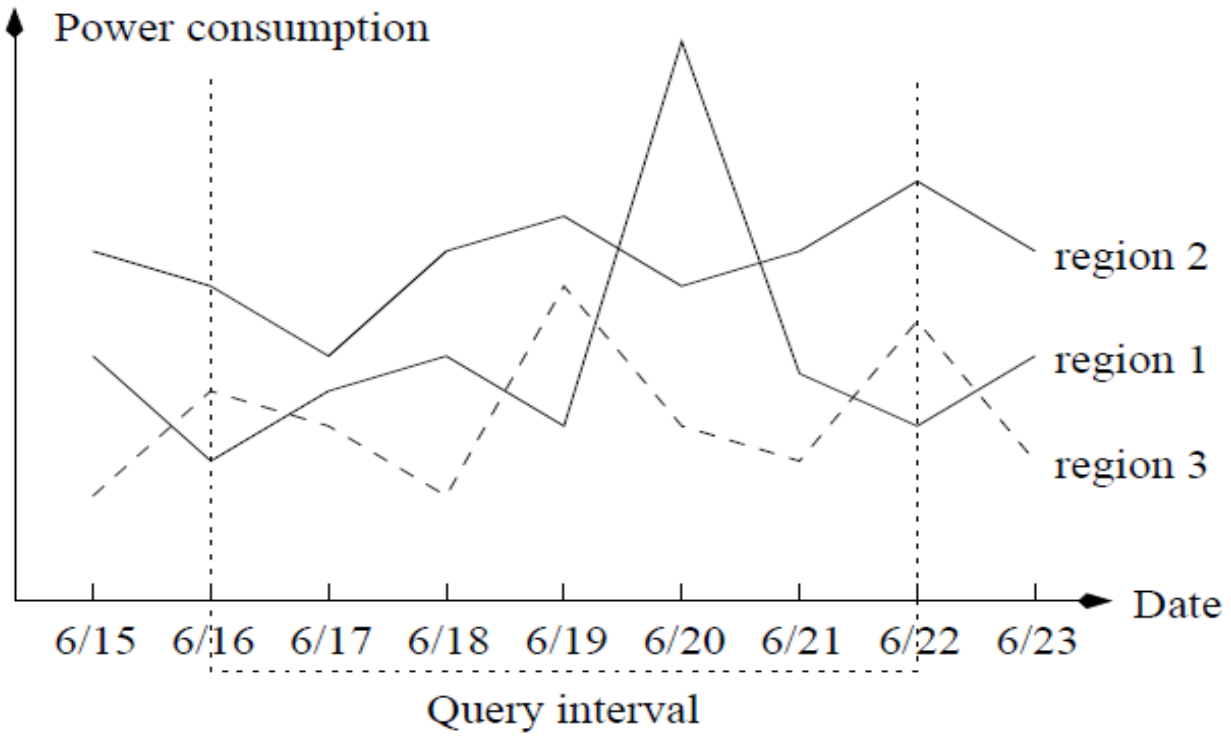


Fig. 1. A set of power consumption time series.

Skyline = {region1, region2}

Intervall-Skyline – Effektive Berechnung

Notation	Meaning
s,q	time series
[i:j](i ≤ j)	an interval
Sky[i:j]	the skyline in interval [i:j]
t_c	the most recent timestamp
w	the size of the base interval
n	the number of time series
$W=[t_c-w+1:t_c]$	the base interval of time series
s.max	the maximum value of s in the base interval W
s.min[i:j]	the minimum value of s in interval [i,j] in W

Gegeben eine Menge von Zeitreihen S, so dass jede Zeitreihe im Basisintervall $W=[t_c-w+1:t_c]$ definiert ist. Gesucht wird eine Datenstruktur D, so dass jede Intervall-Skyline-Anfrage in $[i:j] \subseteq W$ effektiv mit D berechnet werden kann

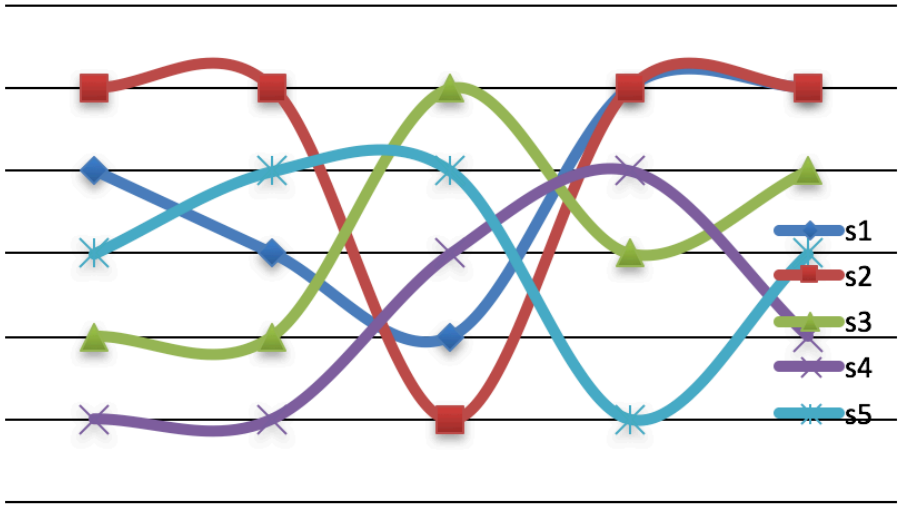
On-the-fly-Methode (OTF)

Online-Intervall-Skyline-Anfragebeantwortung

Beispieldaten:

A SET OF TIME SERIES DATA.

Time series ID	Timestamps					
	1	2	3	4	5	…
s_1	4	3	2	5	5	…
s_2	5	5	1	5	5	…
s_3	2	2	5	3	4	…
s_4	1	1	3	4	2	…
s_5	3	4	4	1	3	…



Intervall-Skyline: OTF-Verfahren

Beh. 1 (max-min)

Für zwei Zeitreihen s, q und Intervall $[i:j] \subseteq W$ gilt:

Falls $s.min[i:j] > q.max$ dann $s \rangle_{[i:j]} q$

Basisidee:

- Speichere s.max und s.min[i:j] für jede Zeitreihe zur Berechnung einer Intervall-Skyline
- Reduziere dadurch ggf. unnötige Prüfungen auf "Dominanz" (aufwendig)

Intervall-Skyline: OTF-Verfahren

Algorithm 1 The on-the-fly query algorithm.

Input: a set S of time series, an interval $[i : j]$;
Output: the skyline in $[i : j]$;
Description:

1: L = a sorted list of the time series in S in the descending order of their $s.max$.
2: $Sky = \emptyset$;
3: $maxmin = -\infty$;
4: let s be the first time series in L;
5: **while** L is not empty and $maxmin \leq s.max$ **do**
6: **if** no time series in Sky dominates s in $[i : j]$ **then**
7: remove the time series from Sky dominated by s in $[i : j]$;
8: $Sky = Sky \cup \{s\}$
9: $maxmin = \max_{q \in Sky}\{q.min[i : j]\}$;
10: **end if**
11: s = the next time series in L;
12: **end while**
13: **return** Sky;

List the “capability” of not being dominated in descending order

Record the maximum “ability” of dominating other time series

if maxmin>s.max,
The remaining time series must be dominated by some Skyline in Sky because:
(1). L is in descending order of s.max
(2). Beh. 1

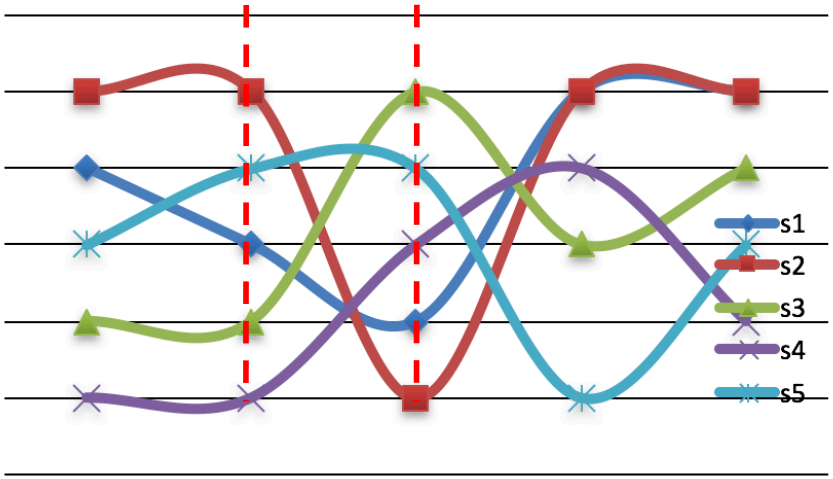
Beispiel

while L is not empty and $maxmin \leq s.max$ **do**
  if no time series in Sky dominates s in $[i:j]$ **then**
    remove the time series from Sky dominated by s in $[i:j]$;
    $Sky = Sky \cup \{s\}$
    $maxmin = \max_{q \in Sky}\{q.min[i:j]\}$;
  end if
  s = the next time series in L;
end while

w=3 W=[1:3]
Berechne Skyline in [2:3]

THE SORTED LIST L IN ALGORITHM 1.

Time series	s_2	s_3	s_5	s_1	s_4
max	5	5	4	4	3
$min[2:3]$	1	2	4	2	1



Sky={}, maxmin = $-\infty$
check s_2, none can dominate s_2, add s_2, maxmin=1
Sky={s_2}
check s_3, none dominates s_3, add s_3, maxmin=2
Sky={s_2, s_3}
check s_5, none dominates s_5, add s_5, maxmin=4
Sky={s_2, s_3, s_5}
check s_1, s_1 is dominated by s_5 , discard, maxmin=4
Sky={s_2, s_3, s_5}
check s_4, maxmin=4 > s_4.max, discard

Ergebnis: Sky ={s_2, s_3, s_5}

Flaschenhals für Online-Verarbeitung?

- Zu berechnende Daten im Verfahren für n Zeitreihen

THE SORTED LIST L IN ALGORITHM 1.

Time series	s_2	s_3	s_5	s_1	s_4
max	5	5	4	4	3
$min[2:3]$	1	2	4	2	1

A SET OF TIME SERIES DATA.

Time series ID	Timestamps					
	1	2	3	4	5	...
s_1	4	3	2	5	5	...
s_2	5	5	1	5	5	...
s_3	2	2	5	3	4	...
s_4	1	1	3	4	2	...
s_5	3	4	4	1	3	...

  - s.min[i:j]
  - s.max für alle Zeitreihen

- Das Fenster bewegt sich (sliding window)
  - Prüfe s.min[i:j] (Teuer? Zeit: O(nw) Platz: O(nw))
  - Prüfe s.max (Teuer? Zeit: O(nw) Platz: O(nw))

Inkrementelle Berechnung

- $maxmin = \max_{q \in Sky}\{q.min[i:j]\};$
- min[i,j] für Zeitreihen in Sky

  Durchsuche [i,j] in der Zeitpunkt-Dimension
  (binärer Suchbaum)

  Gebe min-Wert zurück
  (min Heap)

Vorschlag: Treap (siehe auch: Randomized Binary Search Tree)

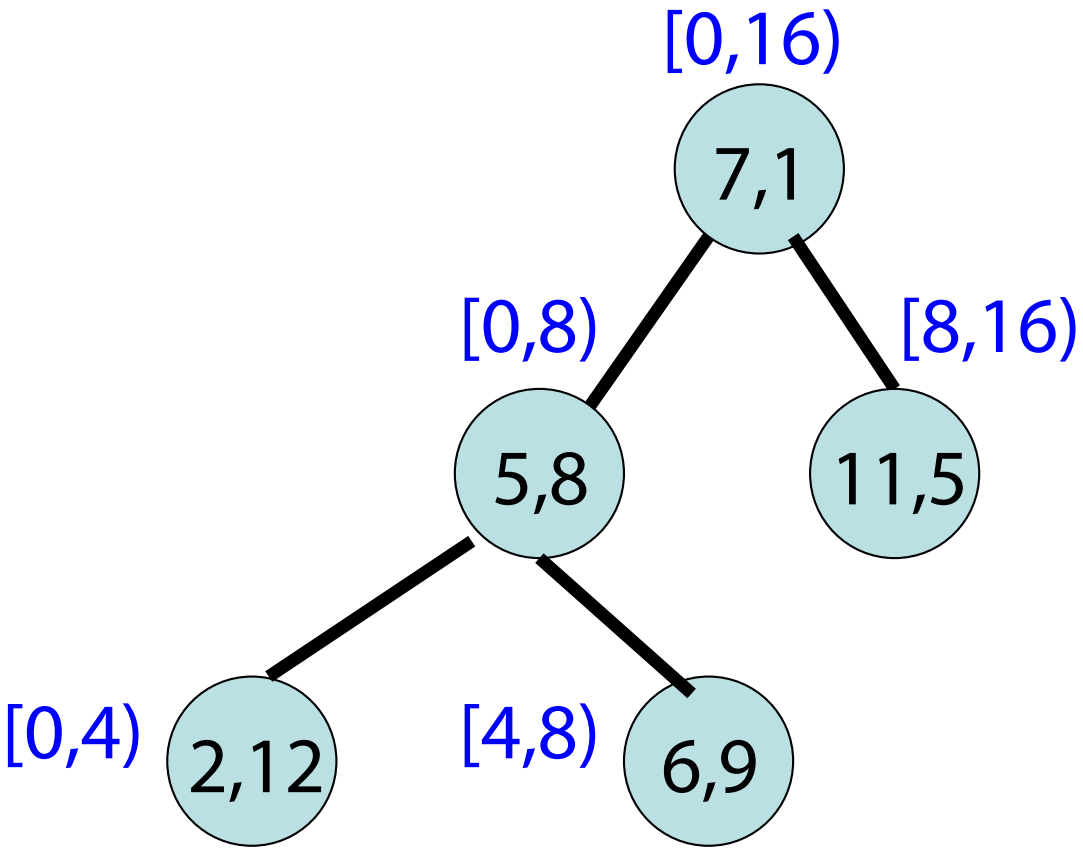
(Verknüpfung von Heap in einer Dimension und binärem Suchbaum in der anderen Dimension)

Aragon, Cecilia R.; Seidel, Raimund, "Randomized Search Trees" (PDF), Proc. 30th Symp. Foundations of Computer Science (FOCS 1989), Washington, D.C.: IEEE Computer Society Press, pp. 540–545, **1989**

Treap

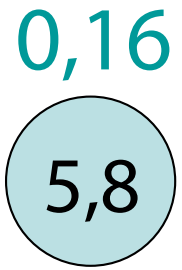
- Daten sind Punkte (x,y)
  → Für Zeitreihen: (Zeitpunkt, Datenwert)
- Alle x Werte sind verschieden, ganzzahlig und liegen im Bereich [0, k)
- Jedem Knoten des Suchbaums ist genau ein Element (x,y) und ein Intervall aus [0, k) zugeordnet
- Der y Wert eines Elements im Knoten w ist ≤ zum y Wert aller Elemente im Unterbaum von w (y Werte definieren einen Min-Baum)
- Das Wurzelintervall ist [0,k)
- Intervall für Knoten w ist [a,b)
  - Linkes Kind hat Intervall [a, floor((a+b)/2))
  - Rechtes Kind hat Intervall [floor((a+b)/2), b)

Beispiel für einen Treap

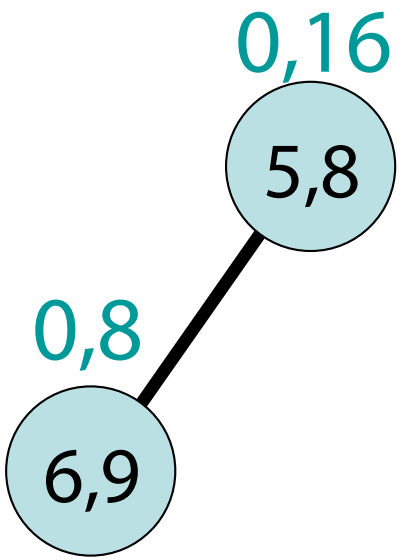


Einfügen

- Starte mit leerem Treap
- k = 16
- Wurzelintervall ist [0,16)
- Füge ein: (5,8)

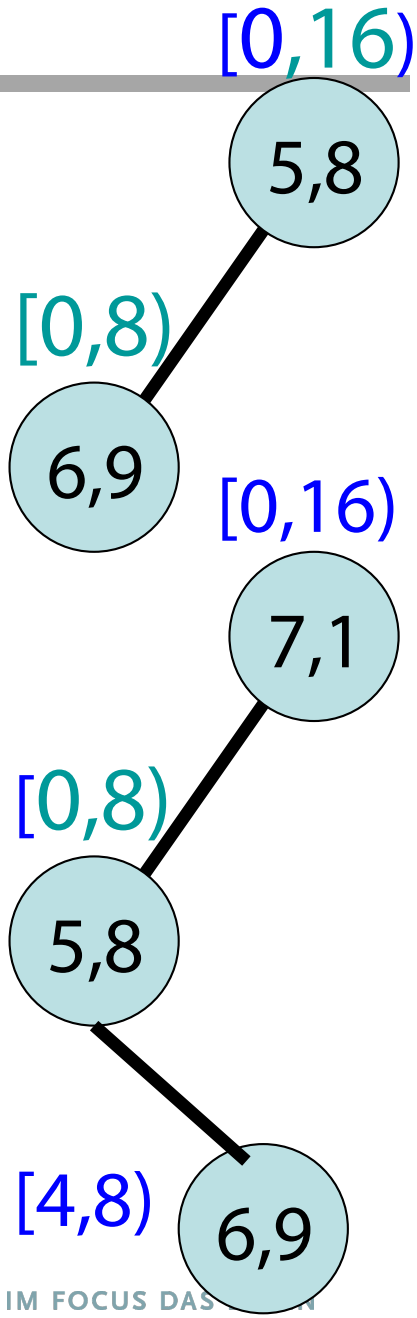


- Füge ein: (6,9)
- (5,8) bleibt Wurzel, weil 8 < 9.
- (6,9) in linken Unterbaum einfügen, weil 6 im linken Kindintervall liegt

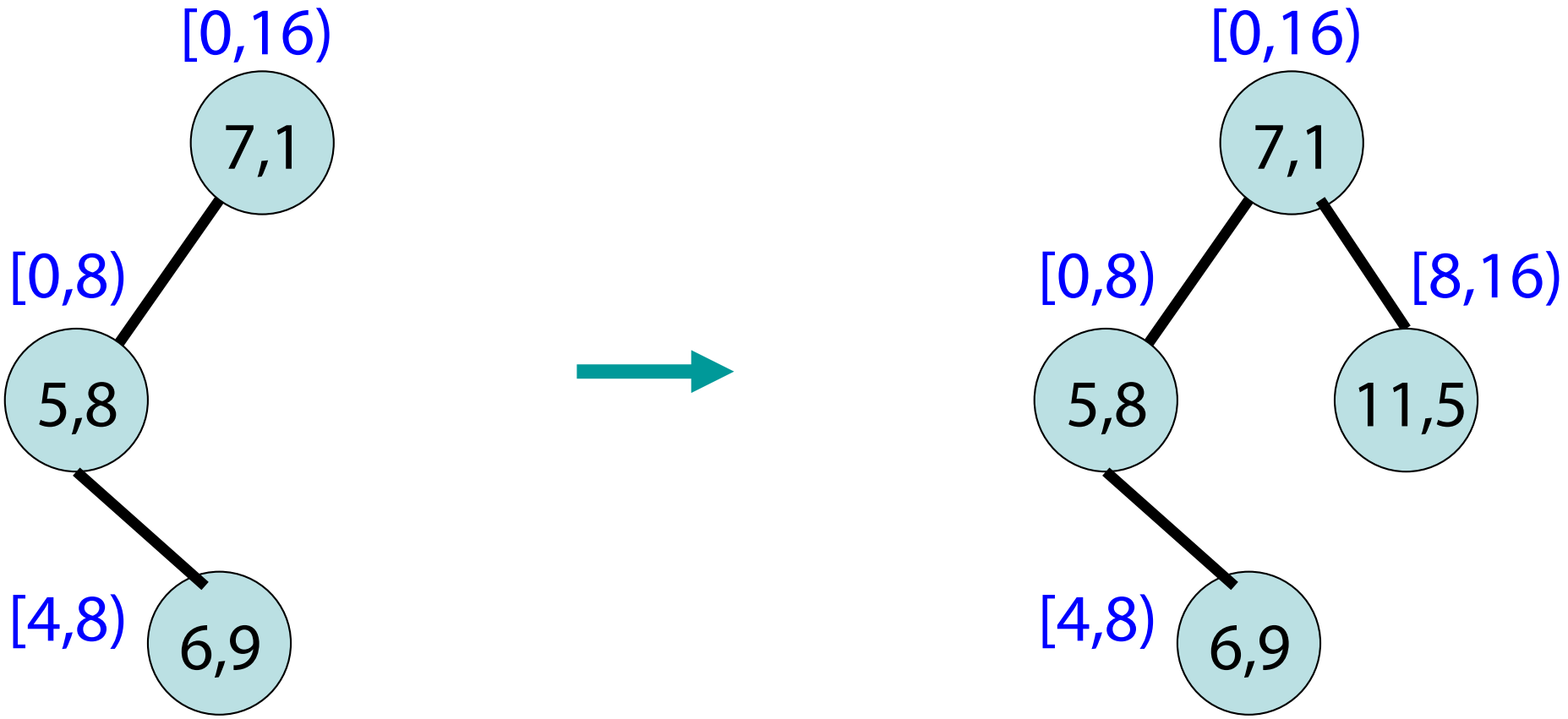


Einfügen

- Füge ein: (7,1)
- (7,1) geht in die Wurzel, weil 1 < 8.
- (5,8) in linken Unterbaum eingefügt, weil 5 ins linke Kindintervall passt
- (5,8) ersetzt (6,9), weil 8 < 9.
- (6,9) in rechten Unterbaum eingesetzt, weil 6 ins rechte Kindintervall gehört

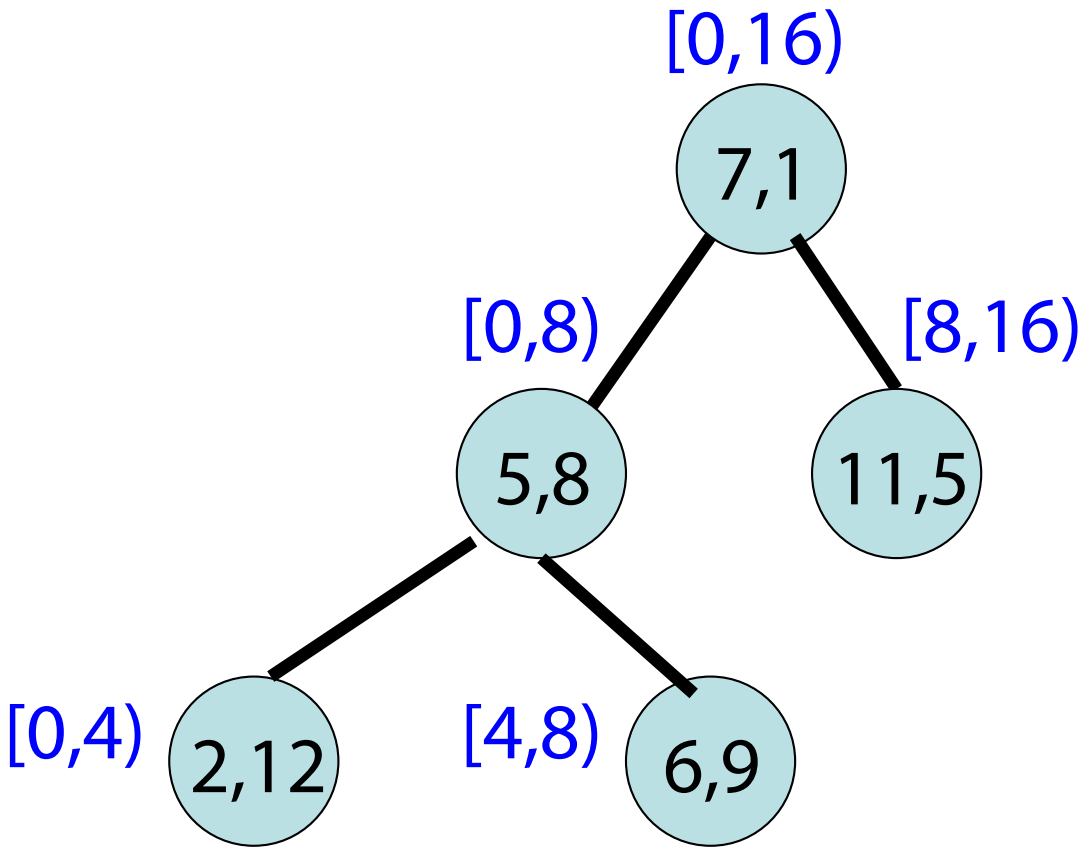


Einfügen



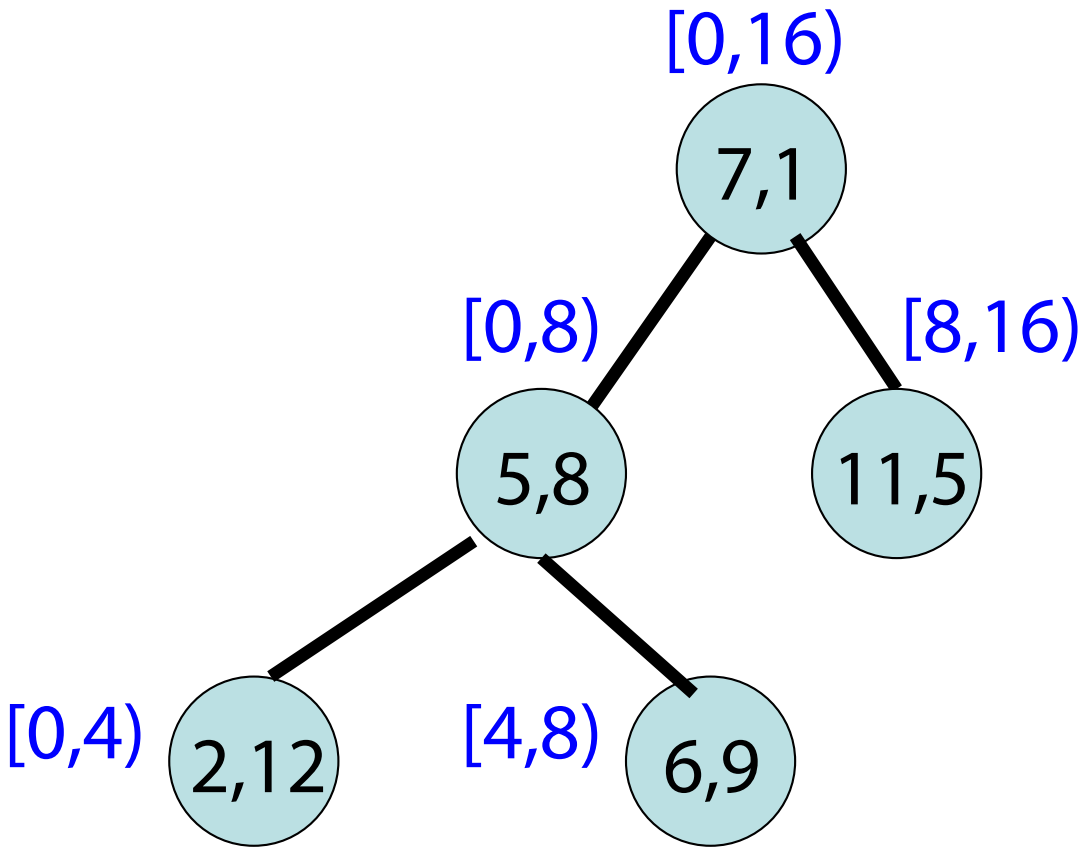
Füge ein: (11,5).

Eigenschaften



- Höhe: O(log k).
- Einfügezeit: O(log k).

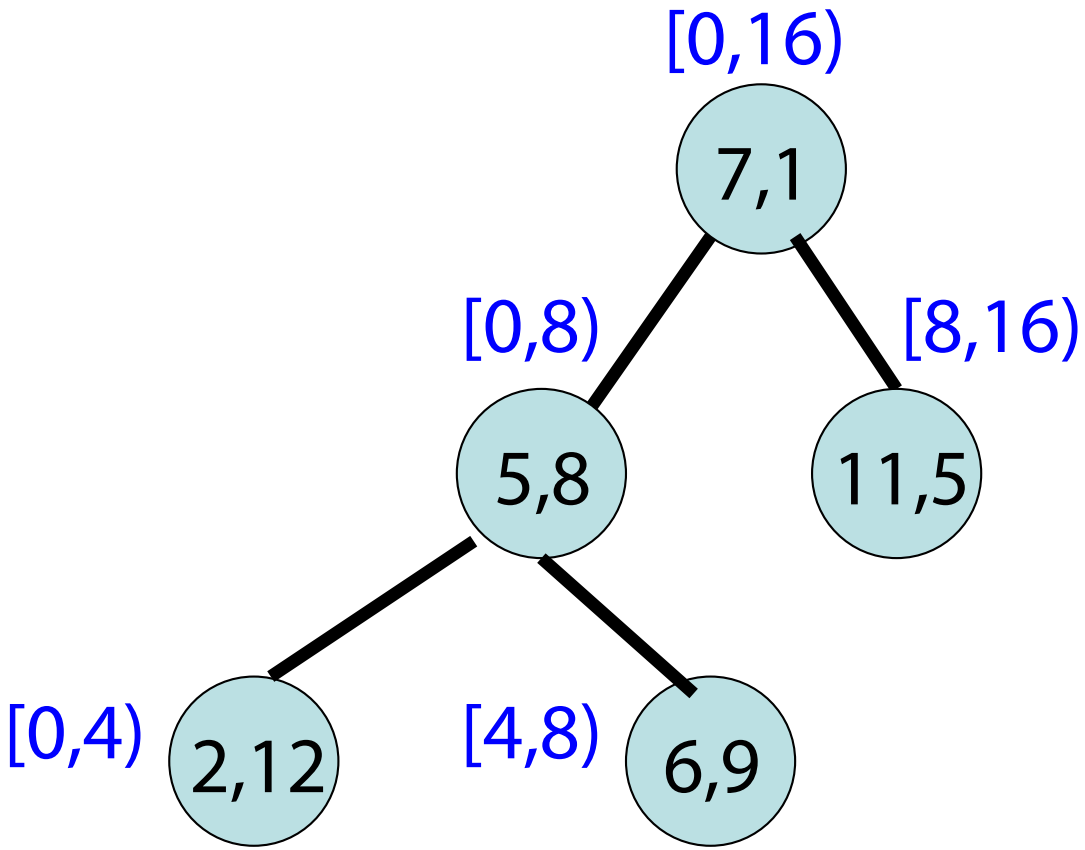
Suche



Suche(1:4)

Suchzeit: O(log k).

Löschen



- Ähnlich zu delete-min beim Min-Heap
- Löschzeit: O(log k).

Inkrementelle Verwaltung von Treaps für Zeitreihen

- Verwende Zeit als Binärbaumdimension (X)
- Verwende Datenwert als als Heap-Dimension (Y)

Basisintervallgröße w ist fix, daher wird W in festen Wertebereich von X abgebildet: {0,1…,w-1}

Die Höhe des Treaps ist fix und balanciert also O(log w)

Einfügen: O(log w)

Löschen: O(log w)

Suche: O(log w)

Inkrementelle Verwaltung von Treaps für Zeitreihen

- Beispiel $k = w$: Bilde Zeitpunkte auf [0, w) ab:
- $w_t := t_c \bmod w$
- $W = [t_c - w + 1 : t_c]$ wird abgebildet auf
  $w_t+1, w_t+2, \ldots w-1, 0, 1, \ldots, w_t$
- Wenn die Zeit fortschreitet: $t_c := t_c + 1$
  $W = [t_c - w + 1 : t_c]$

Sei $t_c = 3$
$w = 3$
$W = [1:3]$

Dann $t_c = 4$
$W = [2:4]$

- Wenn Abbildung t→value der Skyline für den neuen Zeitpunkt den gleichen Wert liefert, wie für den alten:
  → Treap ändert sich nicht
- Wenn gilt: das neue ersetzt das ausgelaufene Element,
  → Treap muss aktualisiert werden (Normalfall)

----- O(log w)

Beispiel

- W=[1:3] und w=3
  - s_1 abgebildet auf (1, 4), (2, 3) und (3, 2)
- W' = [2 : 4]
  - s_1 abgebildet auf (1, 5), (2, 3) und (3, 2)
  - Y-Wert für X = 1 wechselt von 4 auf 5
  - Punkt (1, 4) rausnehmen und (1, 5) einfügen

A SET OF TIME SERIES DATA.

Time series ID	Timestamps					
	1	2	3	1	5	...
s_1	4	3	2	5	5	...
s_2	5	5	1	5	5	...
s_3	2	2	5	3	4	...
s_4	1	1	3	4	2	...
s_5	3	4	4	1	3	...

Bestimmung von min[i:j]

Gegeben ein Treap, dann min[i:j] effektiv bestimmbar

Abbildung von [i:j] in X-Bereich:

$w_i = i \bmod w$; $w_j = j \bmod w$;

Fall 1: $w_i \leq w_j$

Bestimme den min-Wert im Intervall $[w_i, w_j]$

Fall 2: $w_i > w_j$

Bestimme den min-Wert im Intervall $[0, w_j]$

und $[w_j, w-1]$, nehme den kleinsten

$O(\log w)$

Bestimmung von max bei Fensterverschiebung

1: L = a sorted list of the time series in S in the descending order of their $s.max$.

- Max-Berechnung über (v, t)-Paare für jede Zeitreihe
- Fenster verschiebt sich "wenig"
- Strategie: Verwende Hilfsspeicher (auch Skizze genannt)
  - Ein Paar (v, t) wird gehalten, falls keine anderes Paar (v',t') existiert, so dass v'>=v und t'>t
  - Auf diese Weise werden nur im Mittel nur (log w) Paare gehalten

    Finde maximalen Wert: O(1)

    Aktualisiere Zeitstempel O(log w)

J. L. Bentley, H. T. Kung, M. Schkolnick, and C. D. Thompson, "On the average number of maxima in a set of vectors and applications," *J. ACM*, vol. 25, no. 4, pp. 536–543, **1978**

Bestimmung von max bei Fensterverschiebung

A SET OF TIME SERIES DATA.

Time series ID	Timestamps					
	1	2	3	4	5	...
s_1	4	3	2	5	5	...
s_2	5	5	1	5	5	...
s_3	2	2	5	3	4	...
s_4	1	1	3	4	2	...
s_5	3	4	4	1	3	...

Beispiel: max für s_1

W = [1 : 3], Skizze für s1 enthält {(4,1),(3,2),(2,3)}
W = [2 : 4],
(5,4) aktualisiere Skizze
4>3,5>2→ remove (2,3)
4>2,5>3→ remove (3,2)
4>1,5>4→ remove (4,1)
(5,4) left {(5,4)}

Neusortierung nach max bei Fensterverschiebung

- Maximalwert der Zeitreihen kann sich bei Fensterverschiebung ändern
- Es muss die Liste L neu sortiert werden
- Da üblicherweise Varianz im Zeitintervall klein, werden üblicherweise wenig Inversionen benötigt, um L sortiert zu halten

Algorithm 1 The on-the-fly query algorithm.

Input: a set S of time series, an interval $[i : j]$;
Output: the skyline in $[i : j]$;
Description:

1: L = a sorted list of the time series in S in the descending order of their $s.max$.
2: $Sky = \emptyset$;
3: $maxmin = -\infty$;
4: let s be the first time series in L;
5: **while** L is not empty and $maxmin \leq s.max$ **do**
6: **if** no time series in Sky dominates s in $[i : j]$ **then**
7: remove the time series from Sky dominated by s in $[i : j]$;
8: $Sky = Sky \cup \{s\}$
9: $maxmin = \max_{q \in Sky}\{q.min[i : j]\}$;
10: **end if**
11: s = the next time series in L;
12: **end while**
13: **return** Sky;

Analyse von OTF

- Für jede Zeitreihe (Platz)

  Verwende O(w) Platz für Treap

  Verwende O(log w) Platz für Skizze für max Werte

  ---- Platzbedarf für n Zeitreihen O(nw)

- Für jede Zeitreihe (Zeit)

  Verwende O(log w) Schritte, um Treap zu aktualisieren

  Verwende O(log w) Schritte, um Skizze zu aktualisieren

  ---- Amortisierter Zeitbedarf für n Zeitreihen O(n log w)

View-Materialisierungsmethode (VM)

- Hier nicht behandelt

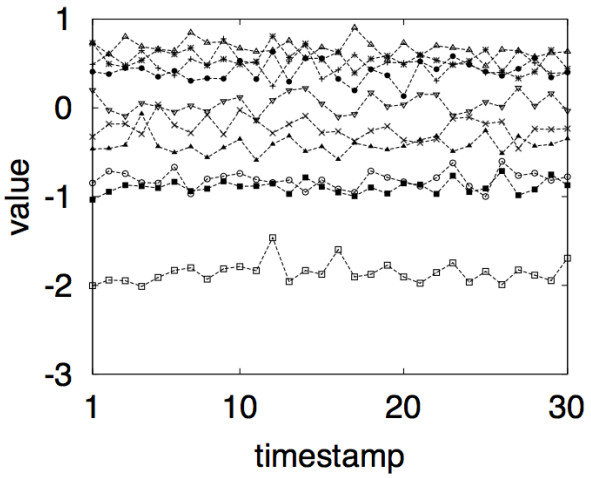
Bin Jiang, Jian Pei, Online Interval Skyline Queries on Time Series,
In Proceedings of the 25th international conference on data engineering (ICDE'09) **2009**

Experimente: Synthetische Datensätze

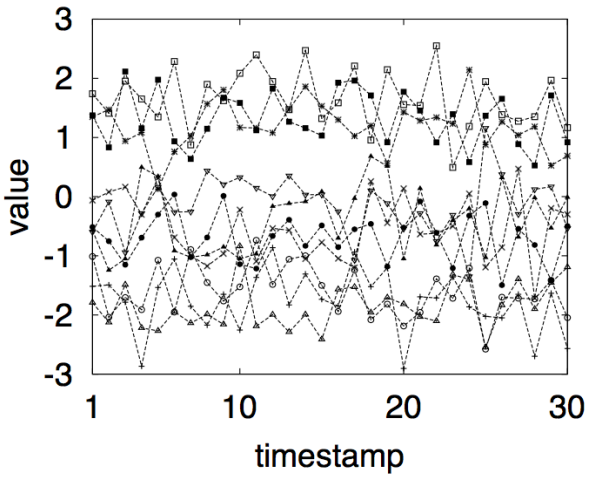
Skyline-Mittelwert μ normalverteilt

Parameter	Values
n	$20k, 40k, \mathbf{60k}, 80k, 100k$
σ	$0.1, 0.3, \mathbf{0.5}, 0.7, 0.9$
W	$100, 200, \mathbf{300}, 400, 500$
w	$50, 100, \mathbf{150}, 200, 250$

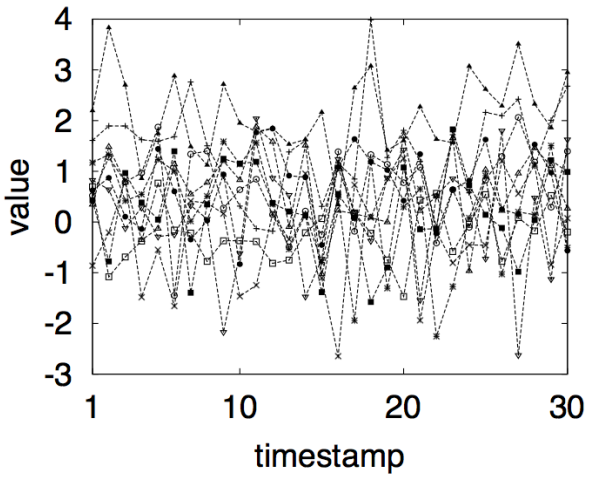
Anzahl Skylines
Varianz
Breite Basisintervall
Breite Anfragefenster



(a) $\sigma = 0.1$.



(b) $\sigma = 0.5$.

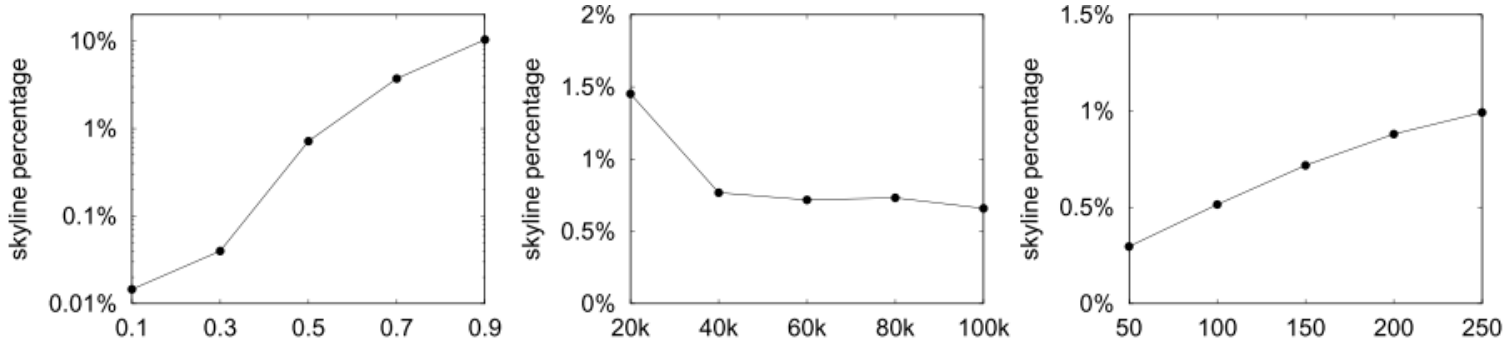


(c) $\sigma = 0.9$.

Bin Jiang, Jian Pei, Online Interval Skyline Queries on Time Series,
In Proceedings of the 25th international conference on data engineering (ICDE'09) **2009**

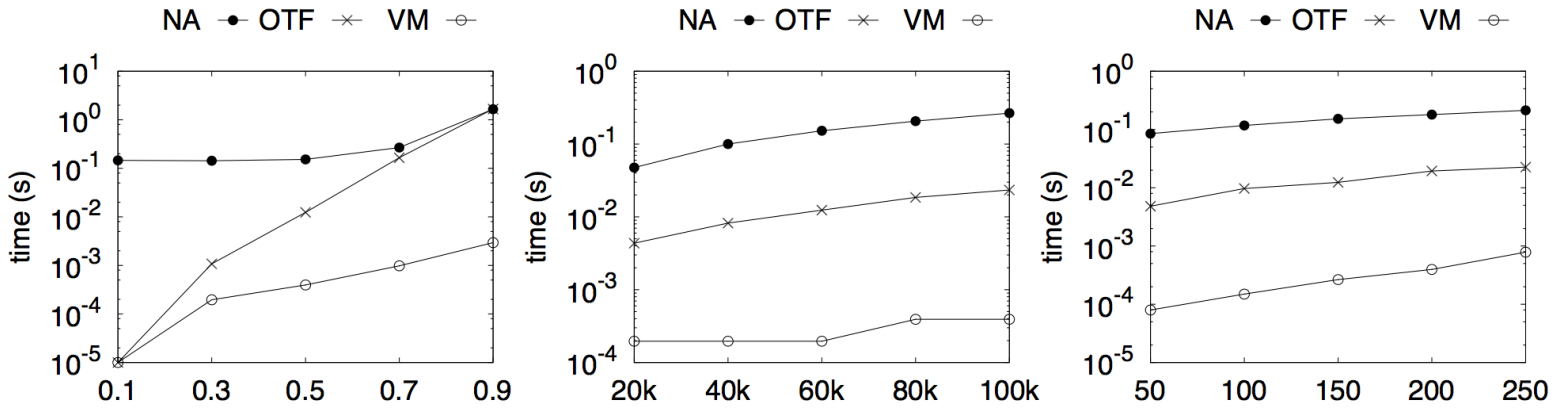
Experimente

- Synthetische Datensätze
  - Anzahl der Skylines



(a) Effect of the standard deviation σ. (b) Effect of the number of time series n. (c) Effect of the interval size w

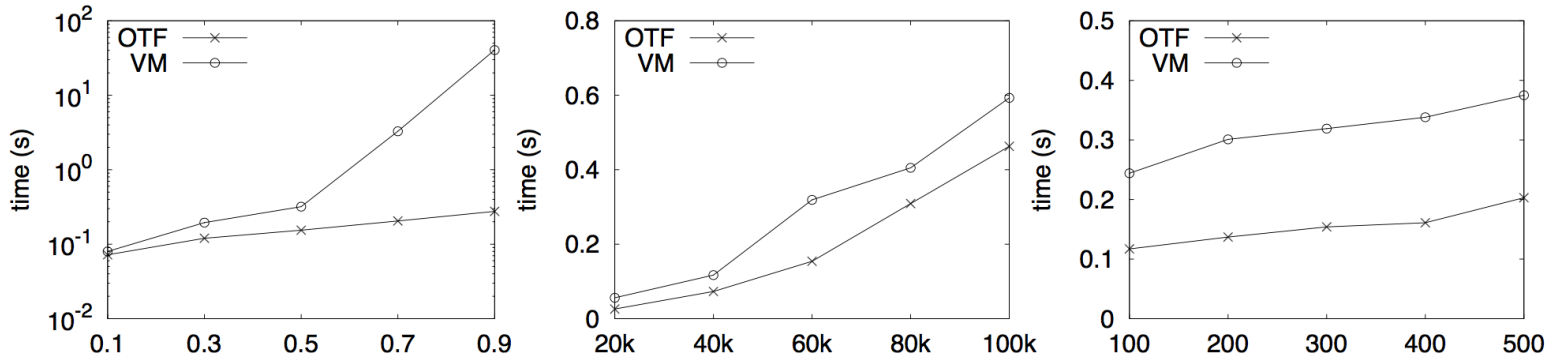
  - Anfragezeit



(a) Effect of the standard deviation σ. (b) Effect of the number of time series n. (c) Effect of the interval size w

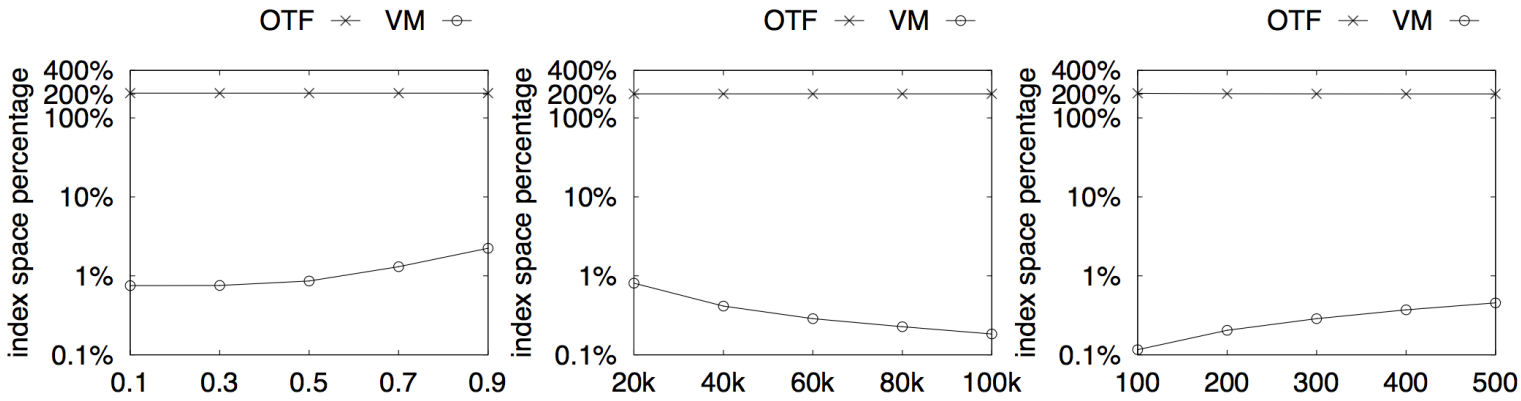
Experimente

- Synthetische Datensätze
  - Aktualisierungseffizienz



(a) Effect of the standard deviation σ. (b) Effect of the number of time series n. (c) Effect of the base interval size W

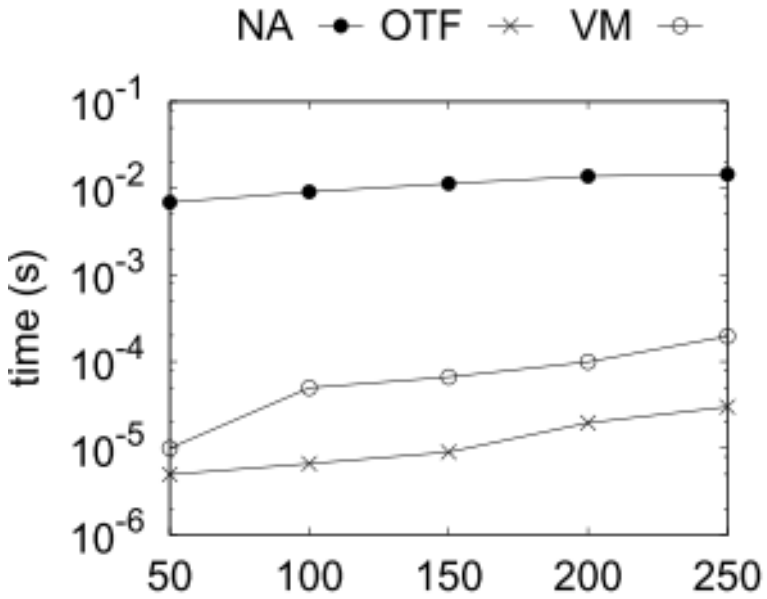
  - Platzbedarf



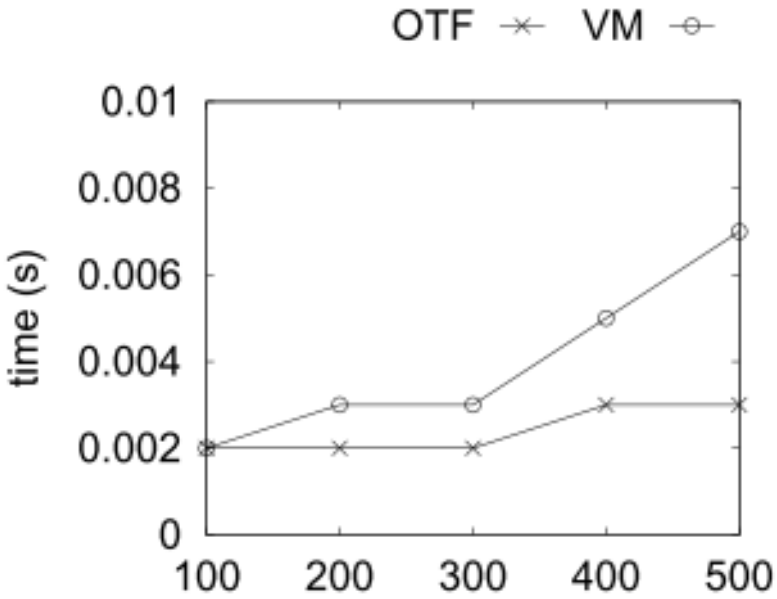
(a) Effect of the standard deviation σ. (b) Effect of the number of time series n. (c) Effect of the base interval size W

Experiments

- Stock Data Sets
  - Query Time



(a) The query time w.r.t. the interval size w



(b) The update time w.r.t. the base interval size w.

Bin Jiang, Jian Pei, Online Interval Skyline Queries on Time Series,
In Proceedings of the 25th international conference on data engineering (ICDE'09) **2009**

Übersicht

Stromkonzept, relationale Anfragesprachen, exakte Auswertung, QoS-Angaben

CQL in Beispielen

No-SQL and NewSQL Datenbanken