Data Streams: Gegenüberstellung existierender Systeme und Architekturen

Marius Renn
Betreuer: Jürgen Göres

Inhalt

- Kriterien
- Existierende Systeme
  - STREAM
  - Aurora
  - PIPES
  - (SPEX)
  - (MAIDS)
- Vergleich
- Fazit

Einleitung
Überblick
Verschiedene Details
Kriterienerfüllung

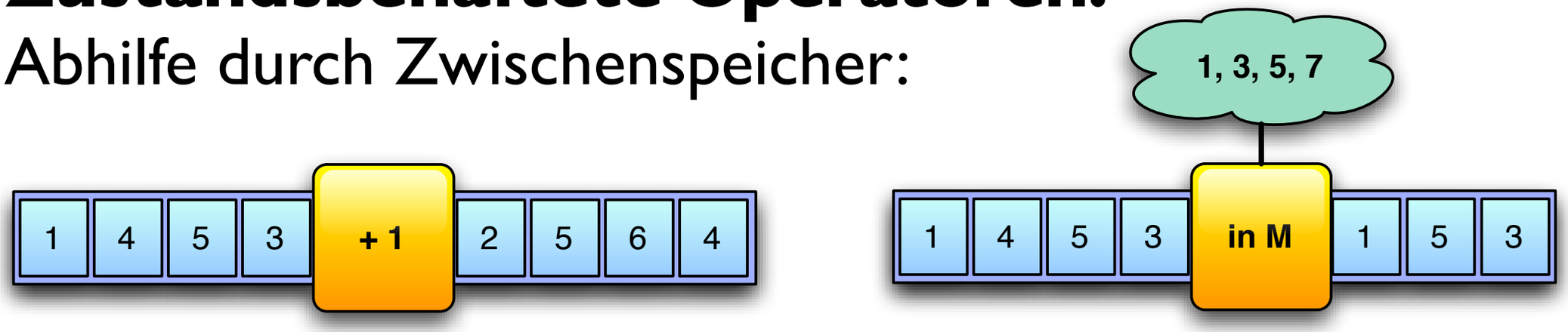
Anfragetypen

Ergebnisausgabe	Definitionszeitpunkt
Einmalig: Liefert ein Ergebnis zu einem Zeitpunkt.	**Vordefiniert:** Anfrage vor Datenverarbeitung bekannt.
Kontinuierlich: Liefert kontinuierliche Ergebnisse.	**Ad-hoc:** Anfrage wird zur Laufzeit gestellt.

→ *Kriterien sind zueinander orthogonal!*

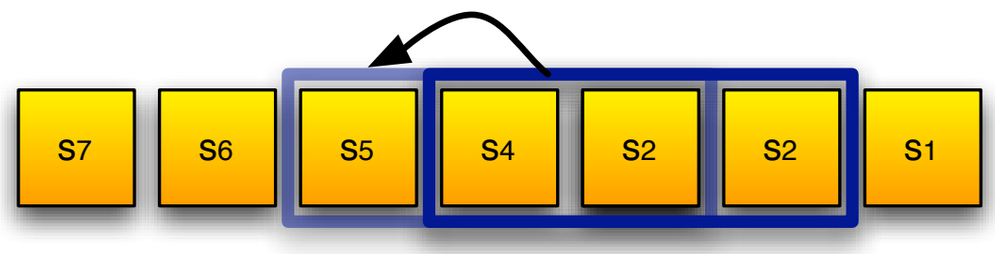
Operatoren

Zustandsbehaftete Operatoren:
Abhilfe durch Zwischenspeicher:

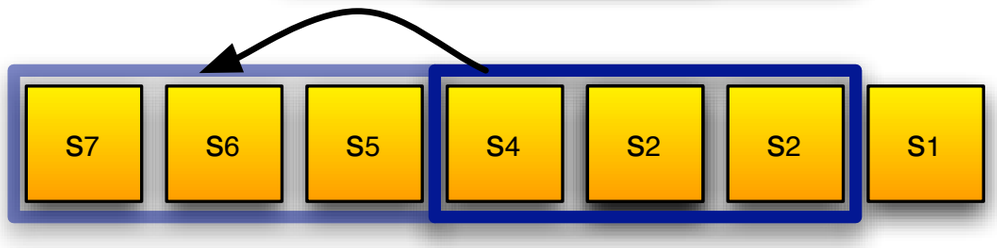


Blockierende Operatoren:
Abhilfe durch (mengen-/zeitbasierte) Fenster:

Sliding Window:



Tumbling Window:



Optimierung und Adaptierung

- **Algebraische Optimierungen** wie in DBVS weiterhin eingschränkt möglich.
- Die Anpassung an die Datenlast heisst **Adaptierung.**
- **Load Shedding:** Auslassen von Tupeln um Last zu verringern.
- **QoS:** Einhaltung von Dienstgütekriterien

STREAM

- **ST**anford St**RE**am Dat**A** **M**anager, entwickelt von der Stanford University, California
- Unterscheidet zwischen Strömen und Relationen.
- 3 Klassen von Operationen: *Relation-zu-Relation*, *Relation-zu-Strom*, *Strom-zu-Relation*
- Anfragesprache **CQL** (continuous query language) als Erweiterung zum populären SQL.

Ströme und Relationen

Strom S

s	t
green	1
red	1
orange	1
blue	2
red	2
black	3
yellow	5
purple	5
red	6

S

Relation R

s
green
red
orange

R(1)

s
blue
red

R(2)

s
black

R(3)

. . .

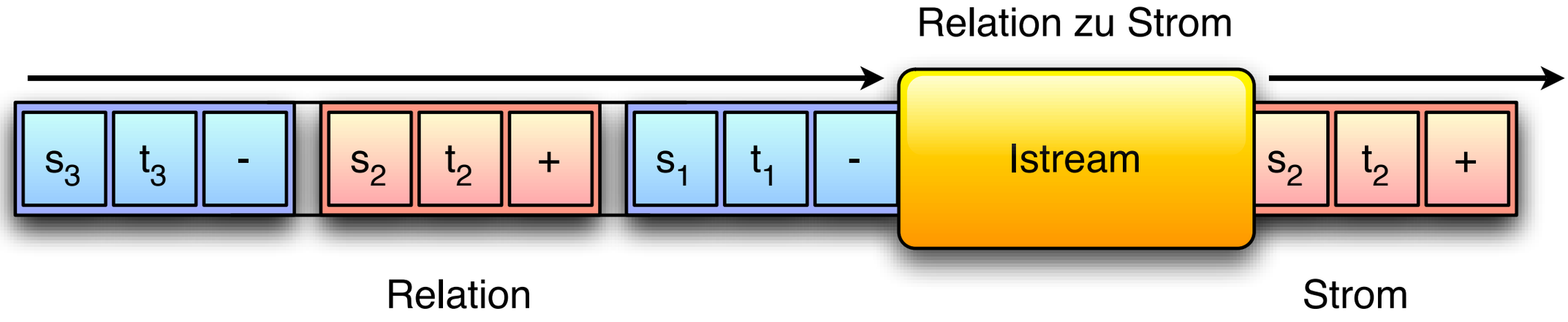
Datenstruktur

"Wie werden Relationen in Ströme verwandelt, und umgekehrt?"

Beiden liegt die gleiche Datenstruktur zugrunde:

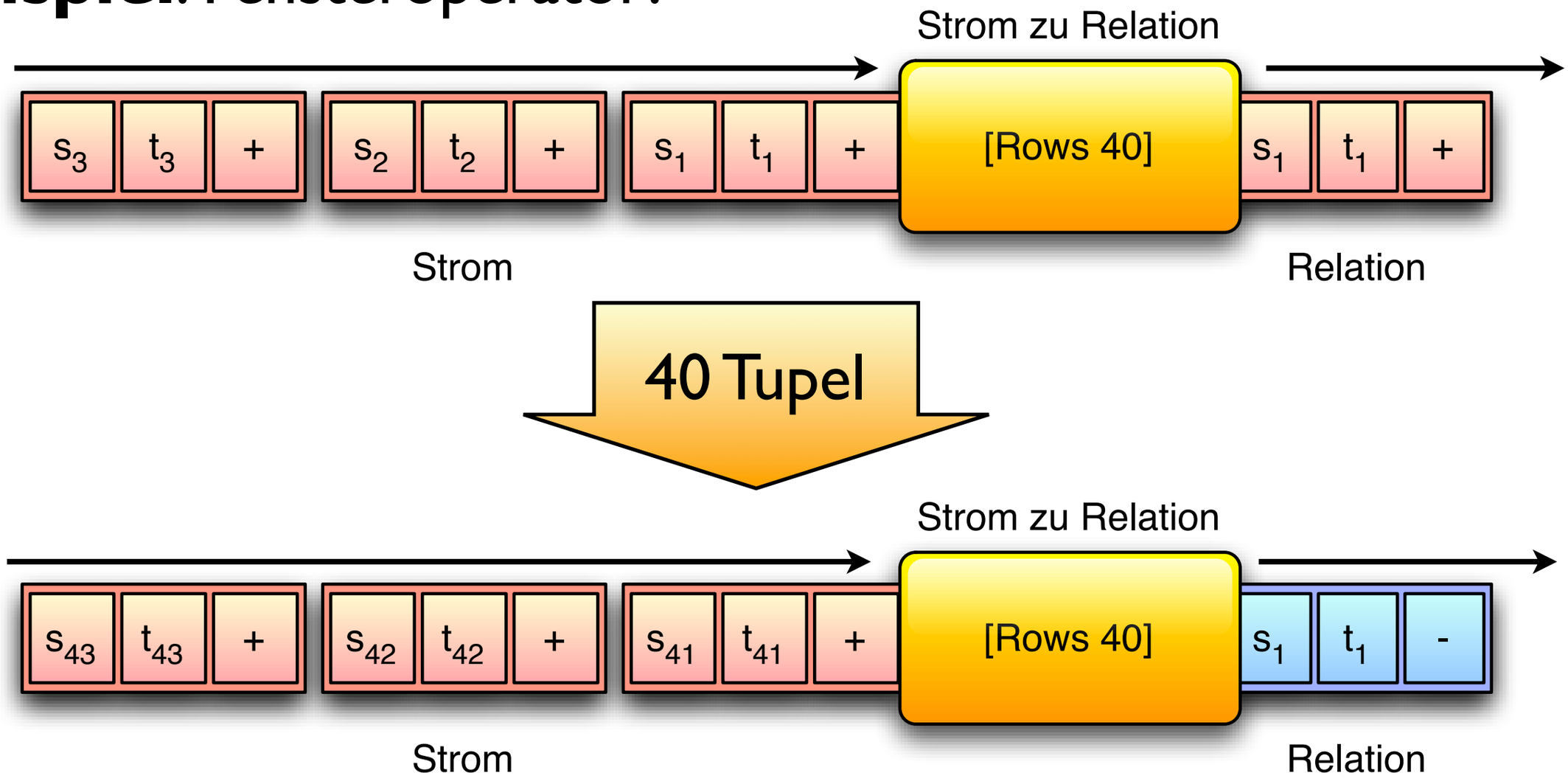
Tupel	Zeitstempel	Operation
data **S**	timestamp **T**	boolean **I**

Ströme besitzen nur positive Elemente, Relationen bestehen aus beiden:



Zustandsbehaftete Operatoren

Beispiel: Fensteroperator:



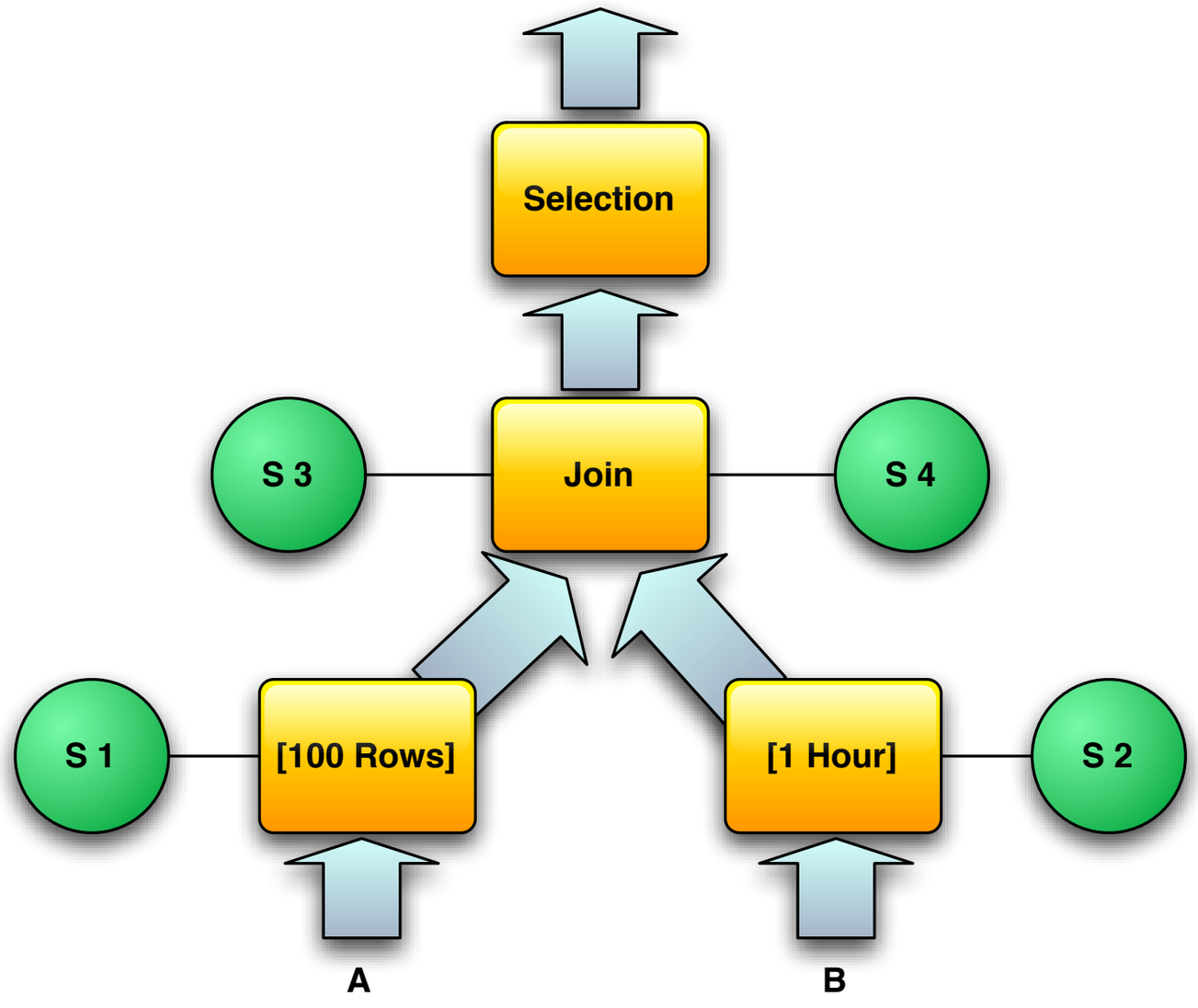
Die zu entferndenden Tripel müssen gespeichert werden!

Synopsen

Synopsen sind den Operatoren zugeteilte **Zwischenspeicher**.

Hohe Redundanz: Oftmals haben mehrere Synopsen ähnlichen Inhalt.

Lösung in STREAM: **Synopsen-Sharing**



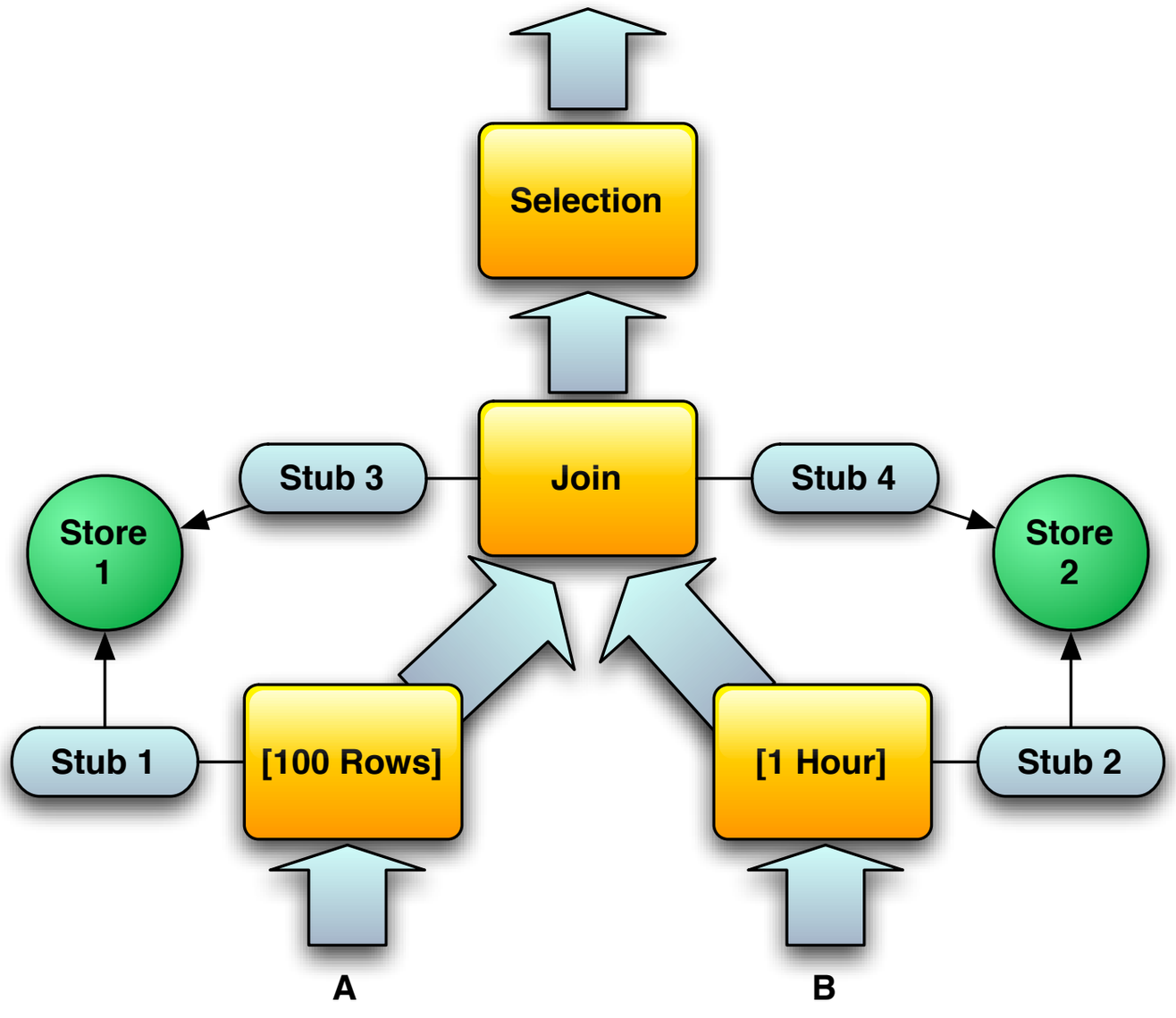
Synopsen-Sharing

Eigentliche Daten liegen in den **Stores**.

Zugriff auf die Daten durch **Stubs**.

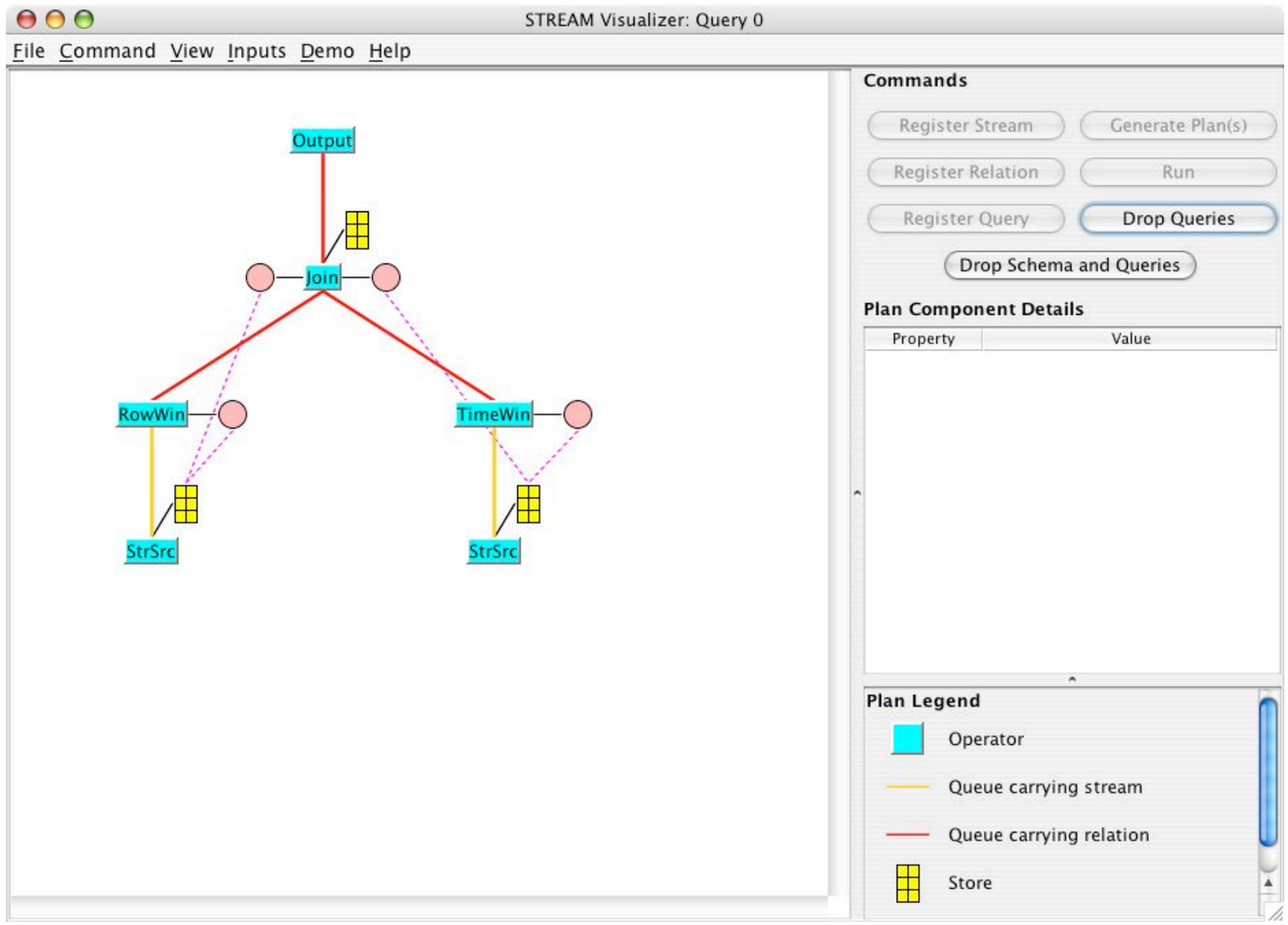
Stubs stellen eine Sicht auf den Store dar.

Aber: Optimierungen beeinflussen evtl. mehrere Operatoren.



STREAM GUI

Um Einblicke in STREAM zu bekommen, liegt ein kostenfreies Testsystem zum Herunterladen bereit.



Kriterien

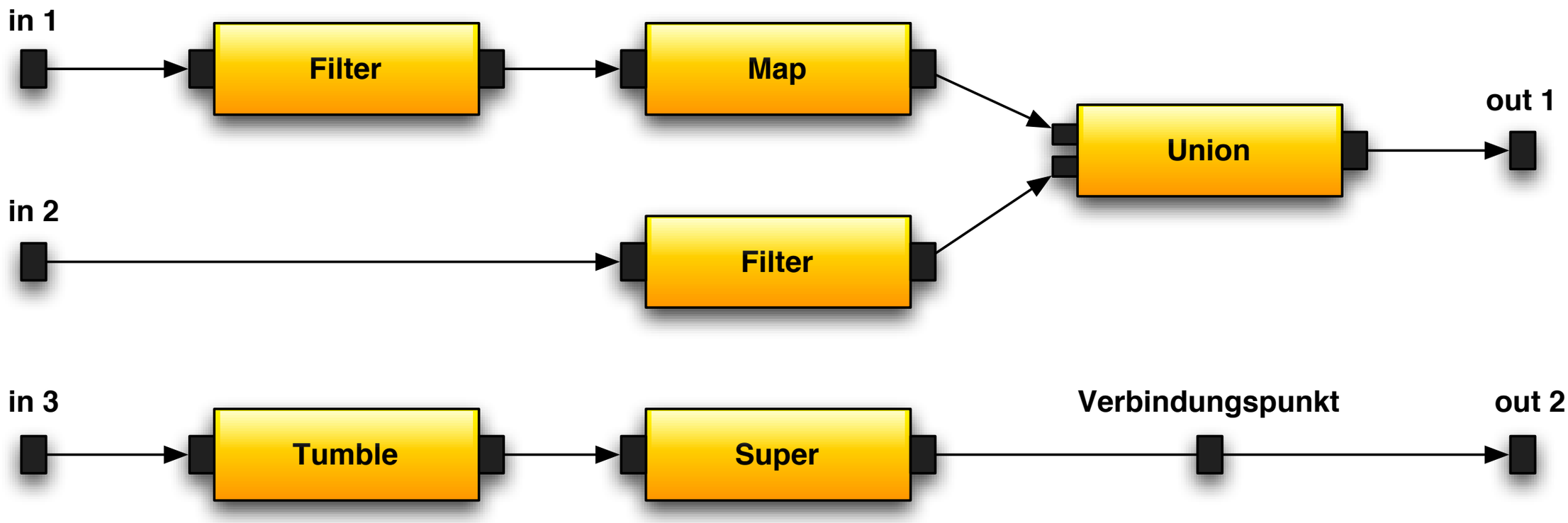
Kriterien	STREAM
Entwickler	Stanford University
Einmalige Anfragen	Ja, auf Relationen
Kontinuierliche Anfragen	Ja, auf Strömen
Vordefinierte Anfragen	Ja
Ad-hoc-Anfragen	Nein
Blockierende Operatoren	Ja, durch Fenster
Zustandsbehaftete Operatoren	Ja, durch Synopsen
Besonderheiten	CQL, unterstützung v. Relationen

Aurora

- Entwickelt von den Universitäten Brown, Brandeis und MIT, alle im Nordosten der USA.
- Einziges hier vorgestelltes kommerzielles System.
- Grafische Anfrageerstellung durch ein System von "**Boxes and Arrows**".
- Reine Stromverarbeitung.
- Unterstützt die Einhaltung von **Dienstgütekriterien**.

Anfragegraphen

Aufbau eines Anfragegraphs in Aurora:

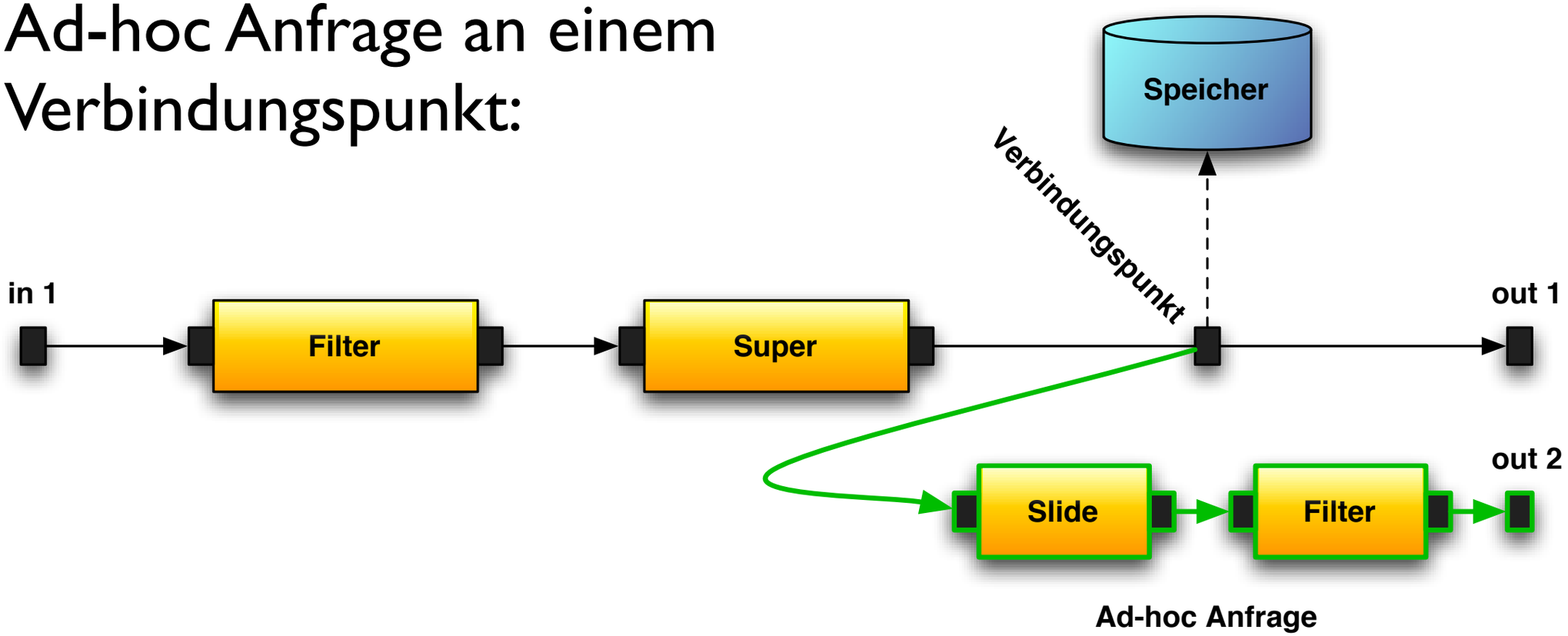


Mehrere unabhängige Teilgraphen möglich!

Beachte: Graphen dürfen keine Zyklen enthalten.

Verbindungspunkte

Ad-hoc Anfrage an einem Verbindungspunkt:

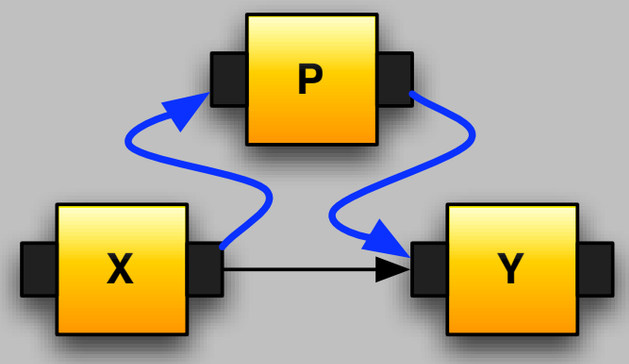


Speicherspezifikation an einem Verbindungspunkt erfolgt durch den Benutzer durch eine **Zeitangabe**.

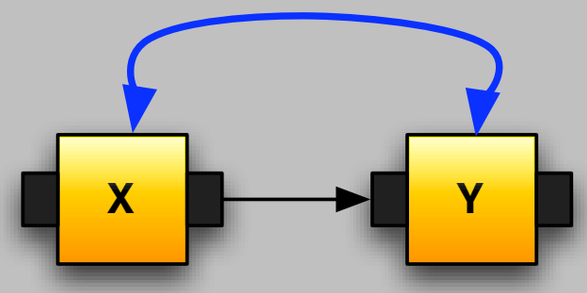
Anfragen zu Daten aus der Vergangenheit sind möglich.

Optimierungsmöglichkeiten

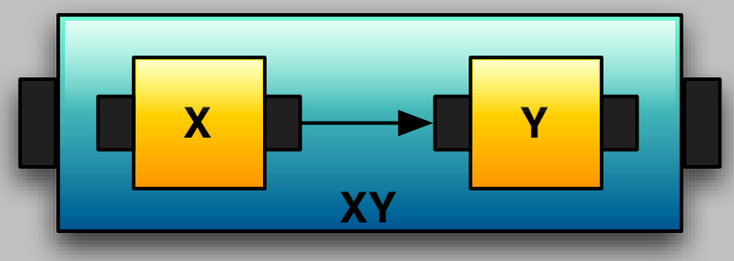
Einfügen v. Projektionen



Umordnen der Boxen

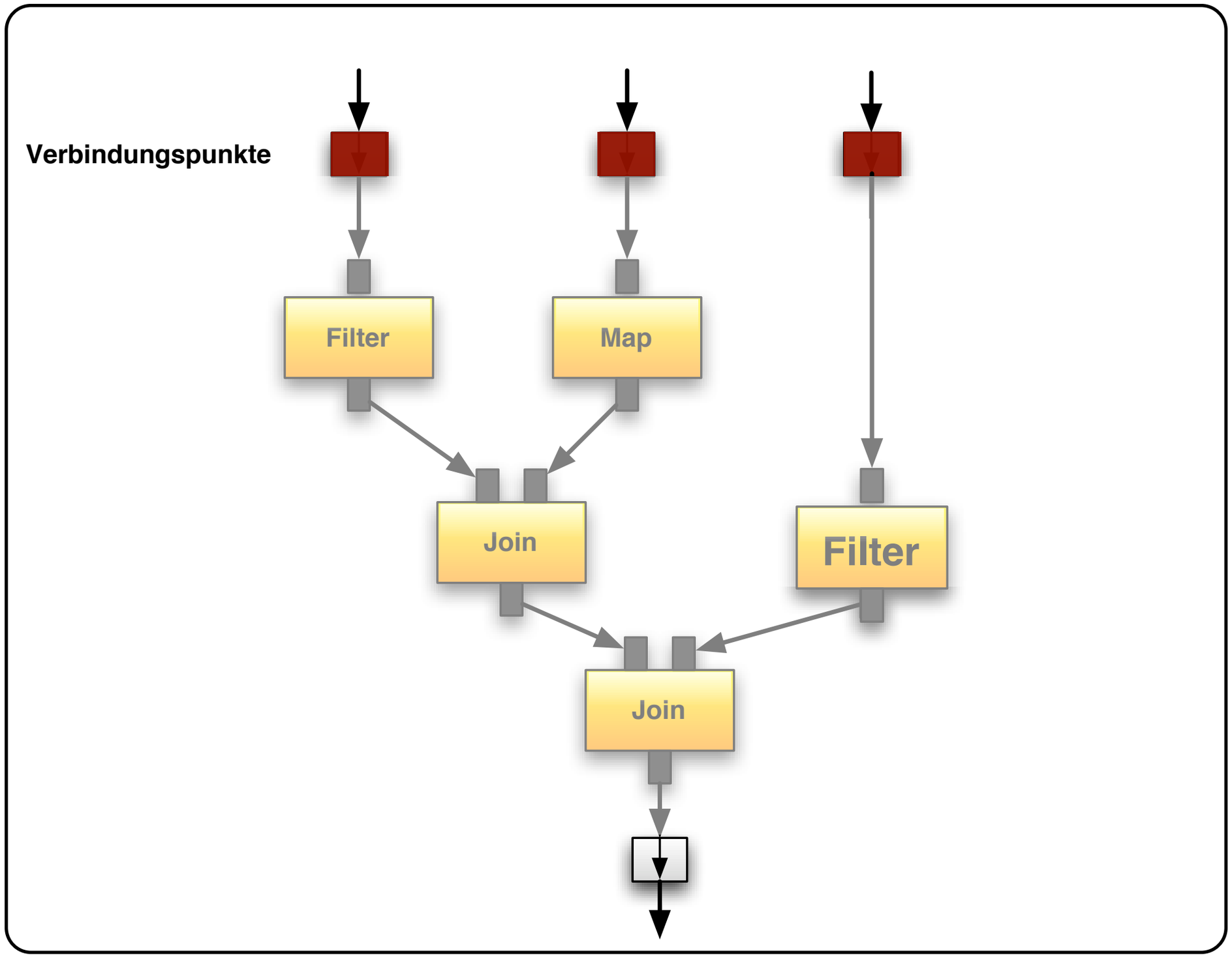


Zusammenfassen v. Boxen



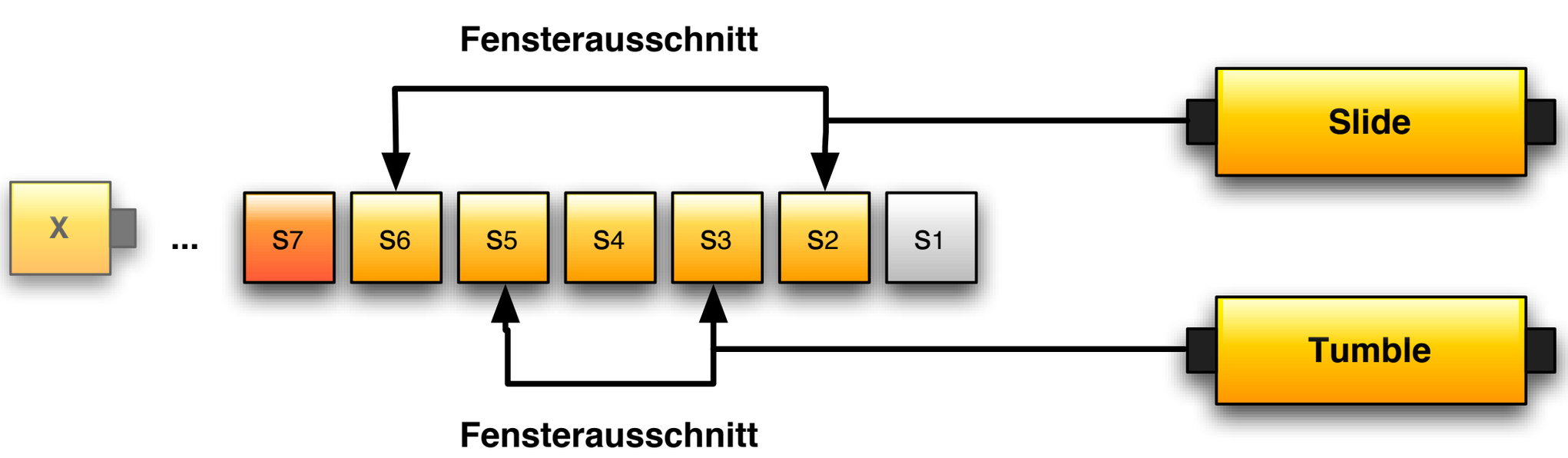
Wie können Teilgraphen zur Laufzeit verändert werden?

Graphoptimierung



Zustandsbehaftete Operatoren

- Aurora macht keinen Unterschied zwischen zustandslosen und -behafteten Operatoren.
- Auf jedem Operator folgt eine **Queue**:

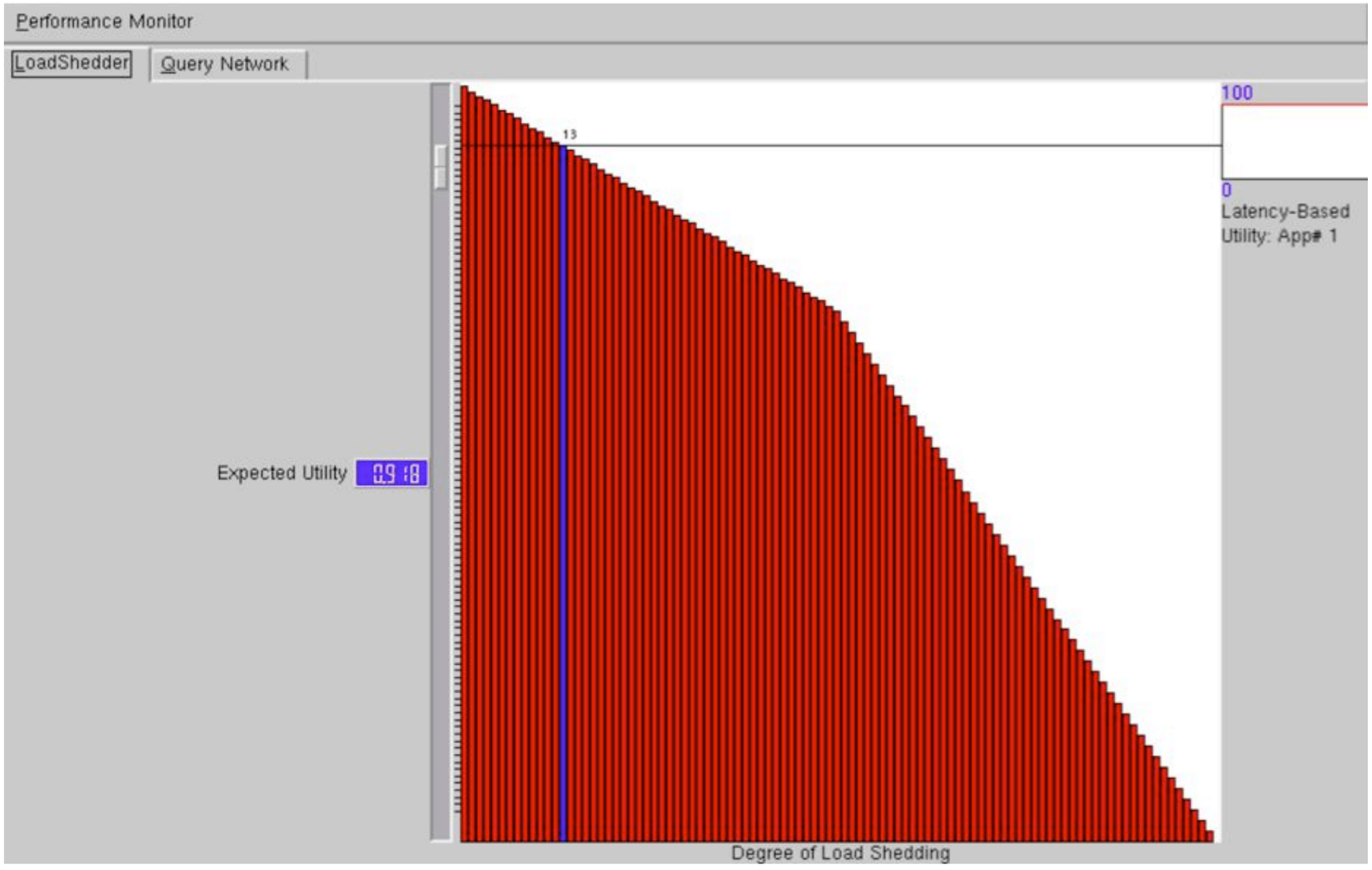


Tupel, die älter sind als die Zugriffsfenster können aus der Queue entfernt werden.

Einhaltung der QoS-Kriterien

"Kann man beobachten, ob die Dienstgütekriterien von Aurora eingehalten werden?"

Ja, umfangreiches Monitoring System von Aurora:



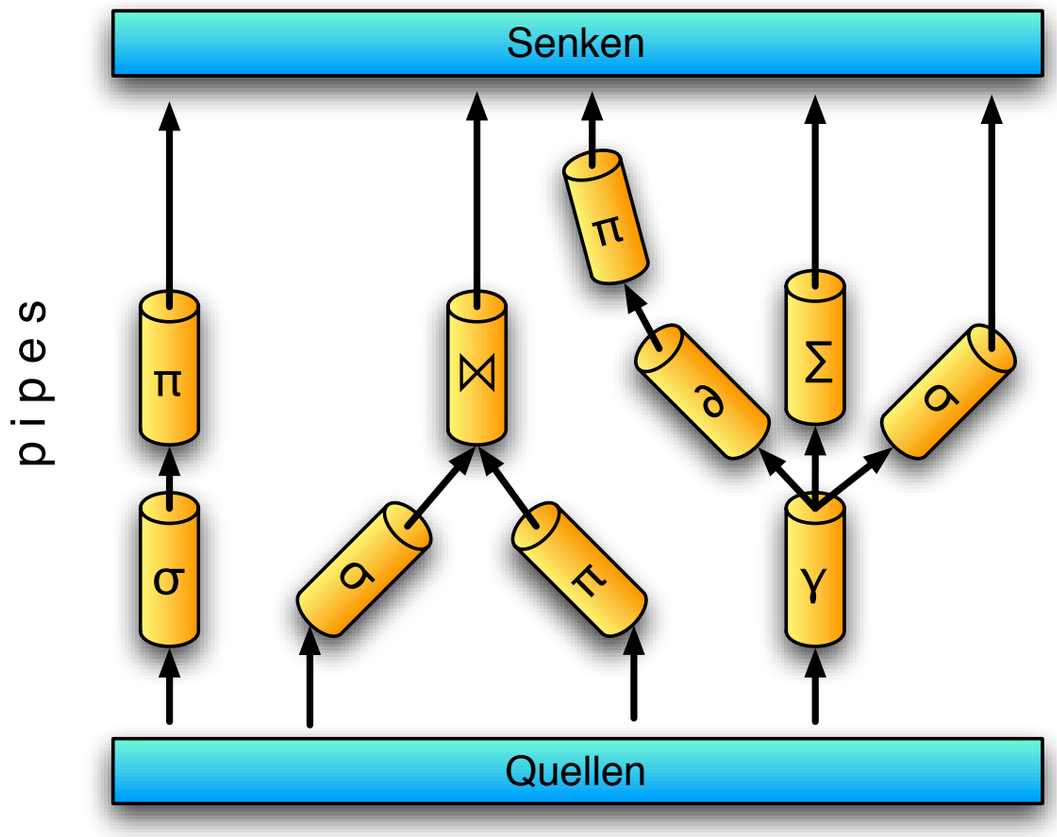
(Aus J. Kern: Aurora Performance Monitoring Tool)

Kriterien

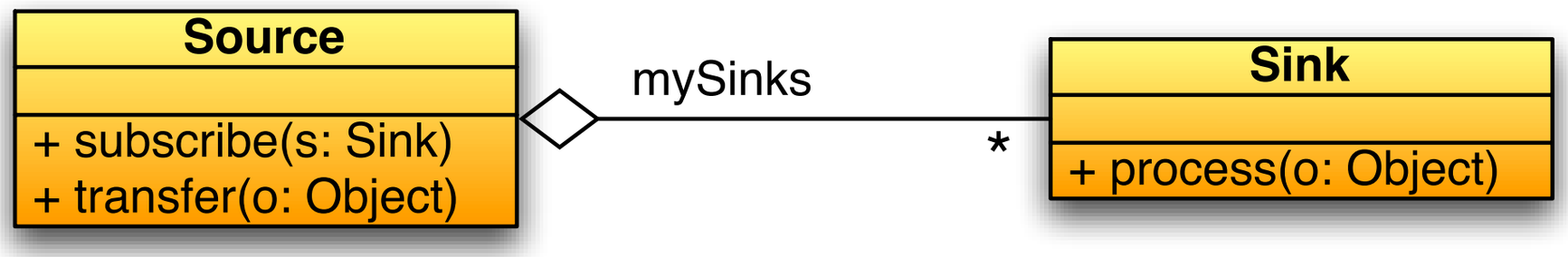
Kriterien	Aurora
Entwickler	Brown, Brandeis, MIT
Einmalige Anfragen	Nein
Kontinuierliche Anfragen	Ja
Vordefinierte Anfragen	Ja
Ad-hoc-Anfragen	Ja
Blockierende Operatoren	Ja, durch Fenster
Zustandsbehaftete Operatoren	Ja, durch Queues
Besonderheiten	QoS-Einhaltung, GUI

PIPES

- **P**ublic **I**nfrastructure for **P**rocessing and **E**xploring **S**treams
- Entwickelt von der Universität Marburg
- Java API
- Implementiert HMTS, Mittelweg zwischen GTS und OTS.



Quellen und Senken



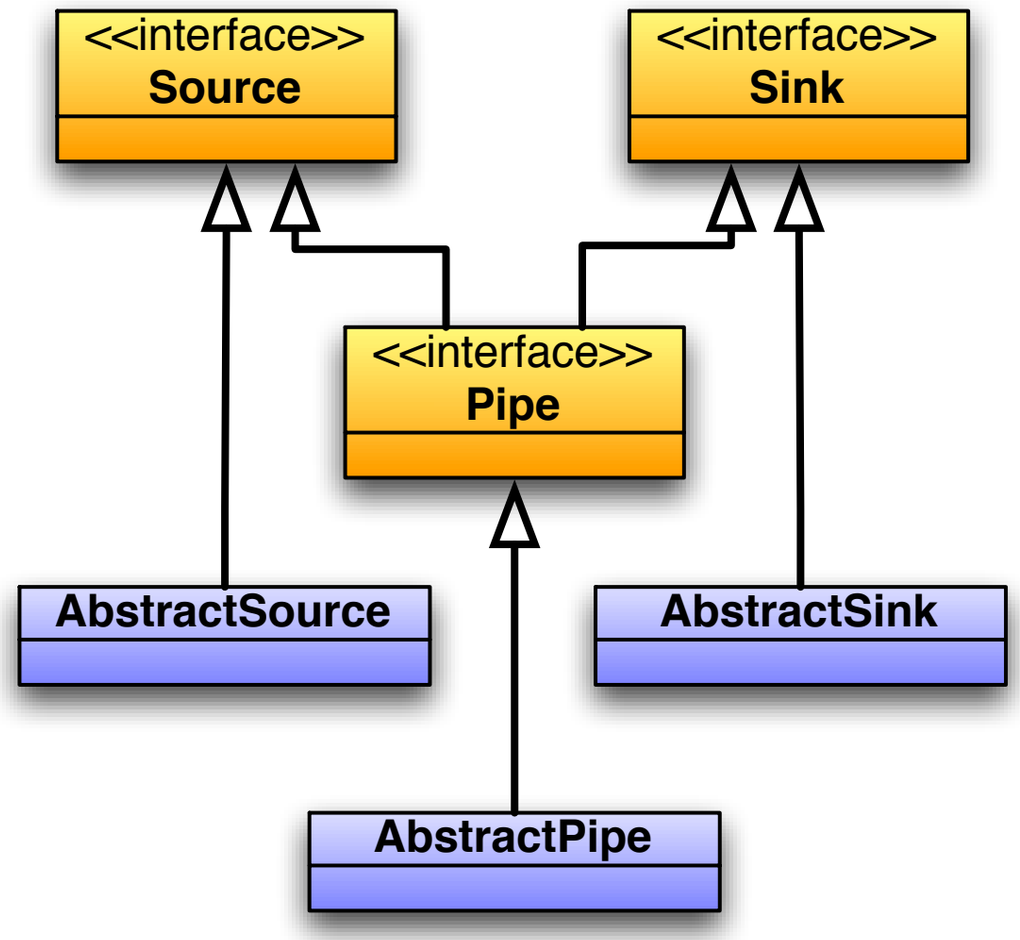

```
void subscribe(Sink sink) {
  mySinks.add(sink);
}
```

```
void transfer(Object o) {
  for(Sink sink: mySinks) {
    sink.process(o);
  }
}
```

PIPES nennt diese Art der direkten Weiterleitung **Direkte Interoperabilität** (direct interoperability).

Klassenhierarchie

- **Pipe** ist sowohl **Quelle** wie auch **Senke**.
- Source, Sink und Pipe sind **Schnittstellen**.
- Abstract-Klassen bieten generische Implementierungen.



Beispiel: Filter

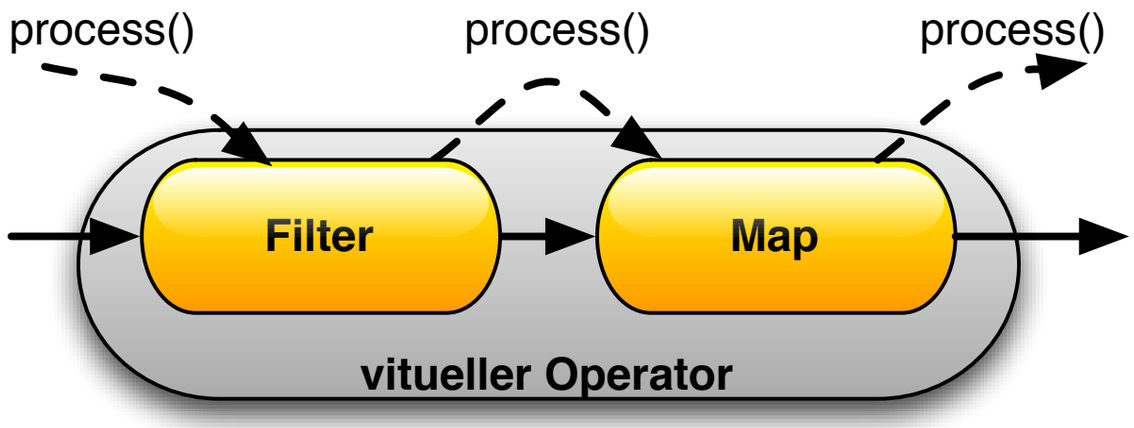
Wir wollen eine Filter-Pipe implementieren:

```
public class Filter extends AbstractPipe
{
   protected Predicate myPred;

   public Filter(Predicate p) {
      myPred = p;
   }

   public void process(Object o) {
      if( myPred.check(o) ) transfer(o);
   }
}
```

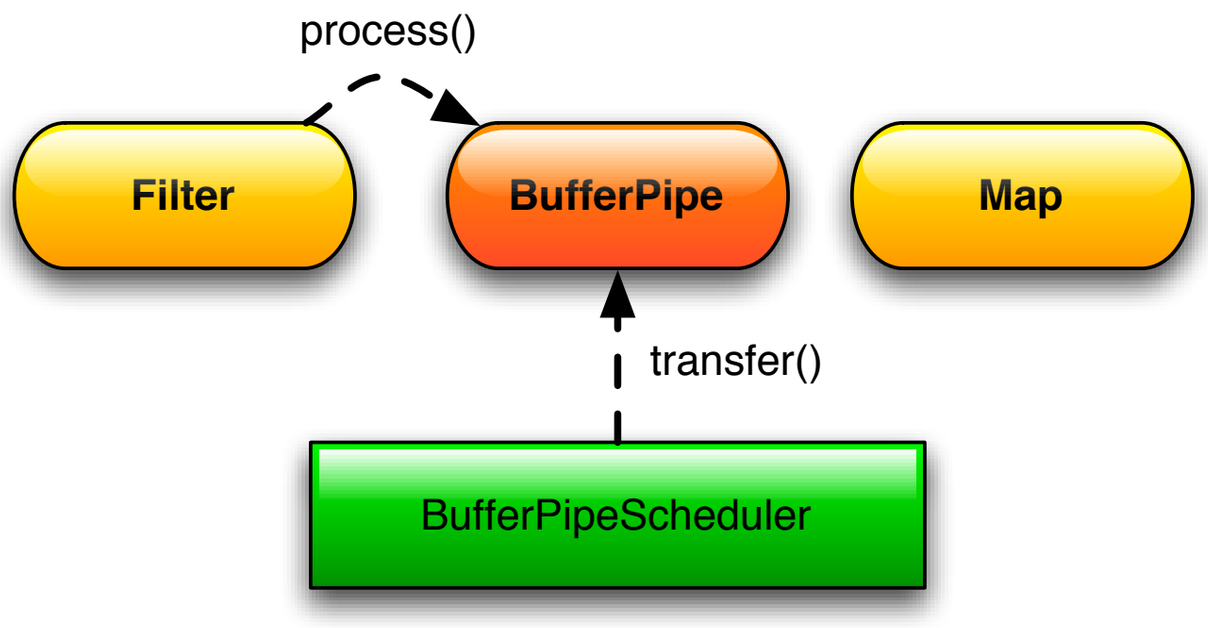
Explizites Scheduling




```
void process(Object o) {
  ...
  transfer(o);
}
```

Pipe

Entkoppelt:

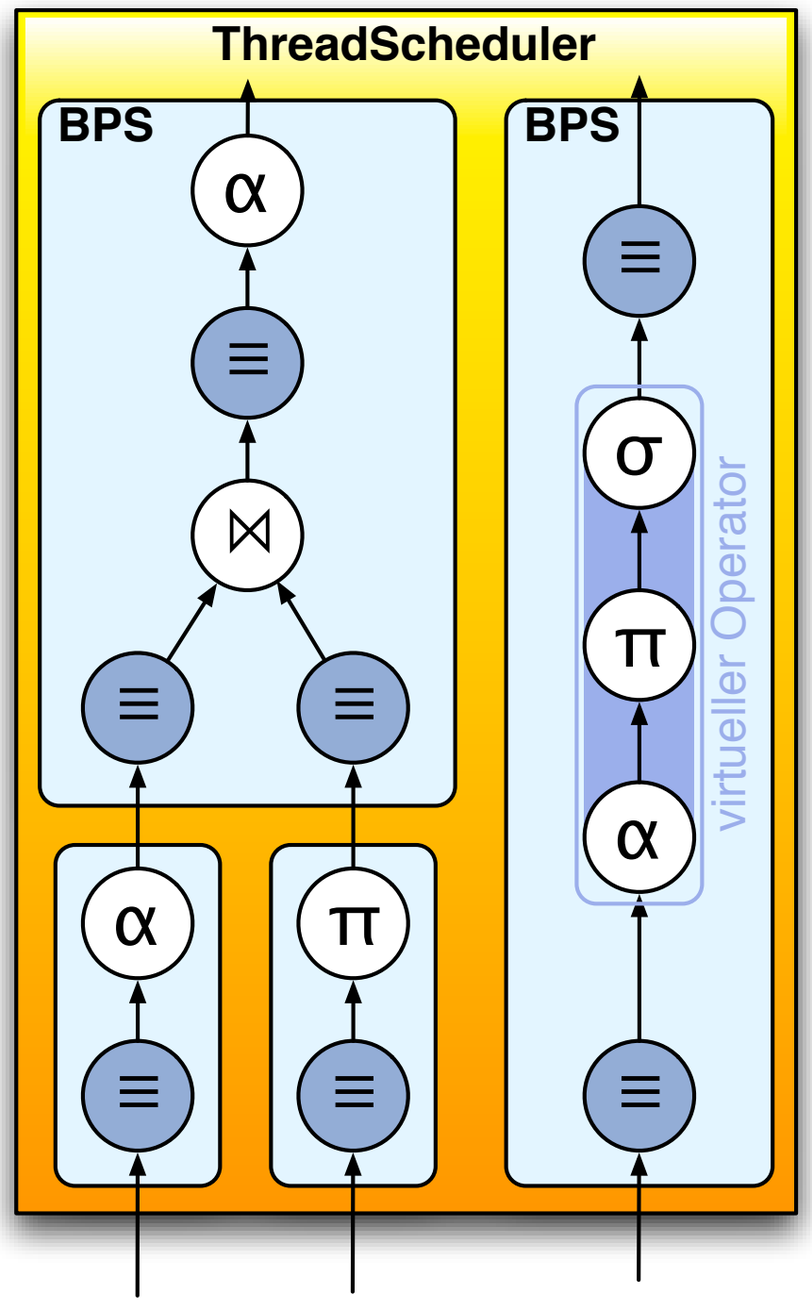



```
void process(Object o) {
  // Add Object to Queue
}
```

BufferPipe

HMTS

- BPS laufen in separaten Threads
- BPS verwalten ihre zugeordneten BPs.
- Direkte Interoperabilität weiterhin möglich
- ThreadScheduler verwaltet Priorität der Threads



Kriterien

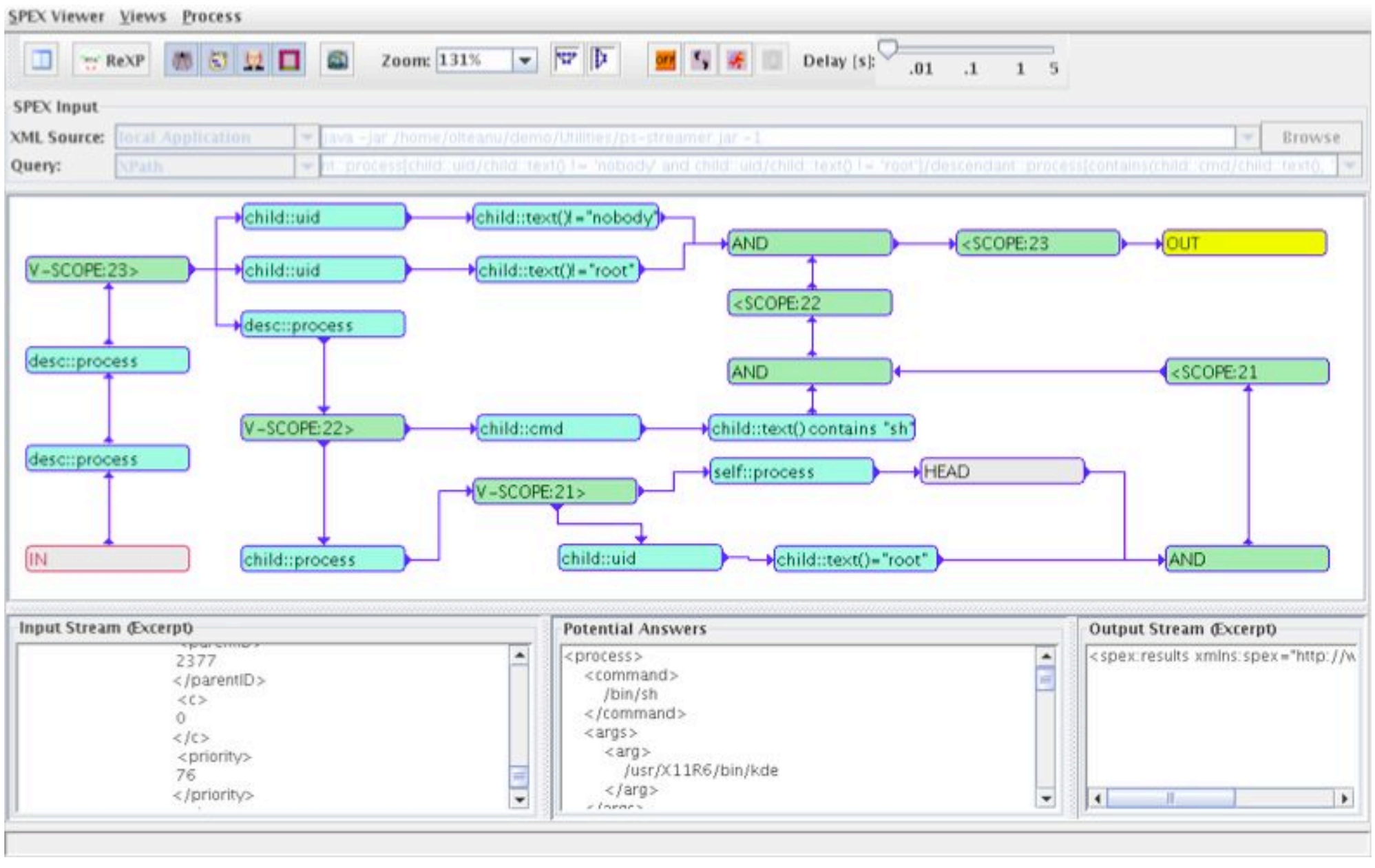
Kriterien	PIPES
Entwickler	Universität Marburg
Einmalige Anfragen	Nein
Kontinuierliche Anfragen	Ja
Vordefinierte Anfragen	Ja
Ad-hoc-Anfragen	Nein
Blockierende Operatoren	Ja, durch SweepAreas
Zustandsbehaftete Operatoren	Ja, durch SweepAreas
Besonderheiten	HMTS

SPEX

- Entwickelt von der Universität München
- Verarbeitet **XML**-Datenströme
- Anfragen werden in **XPath** gestellt
- Beinhaltet eine grafische Benutzeroberfläche
- Konvertierung der XPath Anfrage in einen physischen Anfrageplan erfolgt in 3 Schritten.

SPEX Viewer

SPEX bietet eine grafische Benutzeroberfläche namens *SPEX Viewer*:

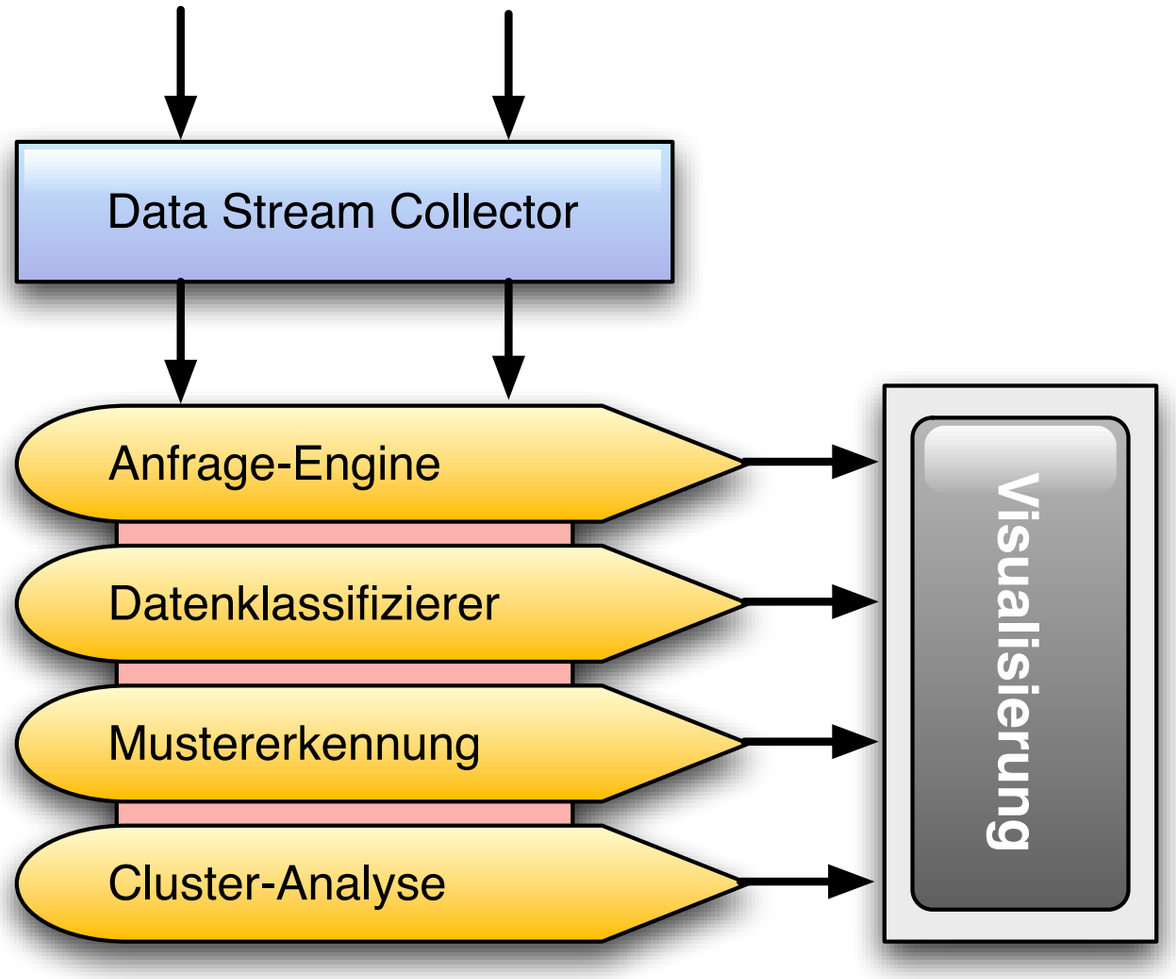


Kriterien

Kriterien	SPEX
Entwickler	Universität München
Einmalige Anfragen	Nein
Kontinuierliche Anfragen	Ja
Vordefinierte Anfragen	Ja
Ad-hoc-Anfragen	Nein
Blockierende Operatoren	-
Zustandsbehaftete Operatoren	-
Besonderheiten	XML-Ströme

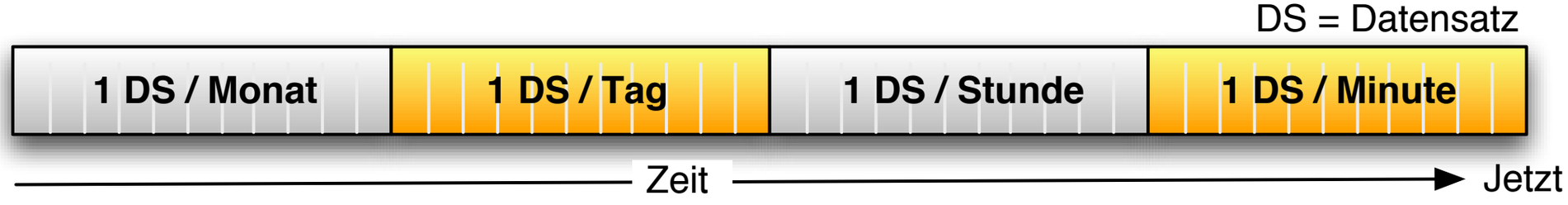
MAIDS

- In der Entwicklung an der University of Illinois
- Soll Data-Mining auf Datenströmen realisieren
- Besitzt mehrere Module um Daten zu verarbeiten.

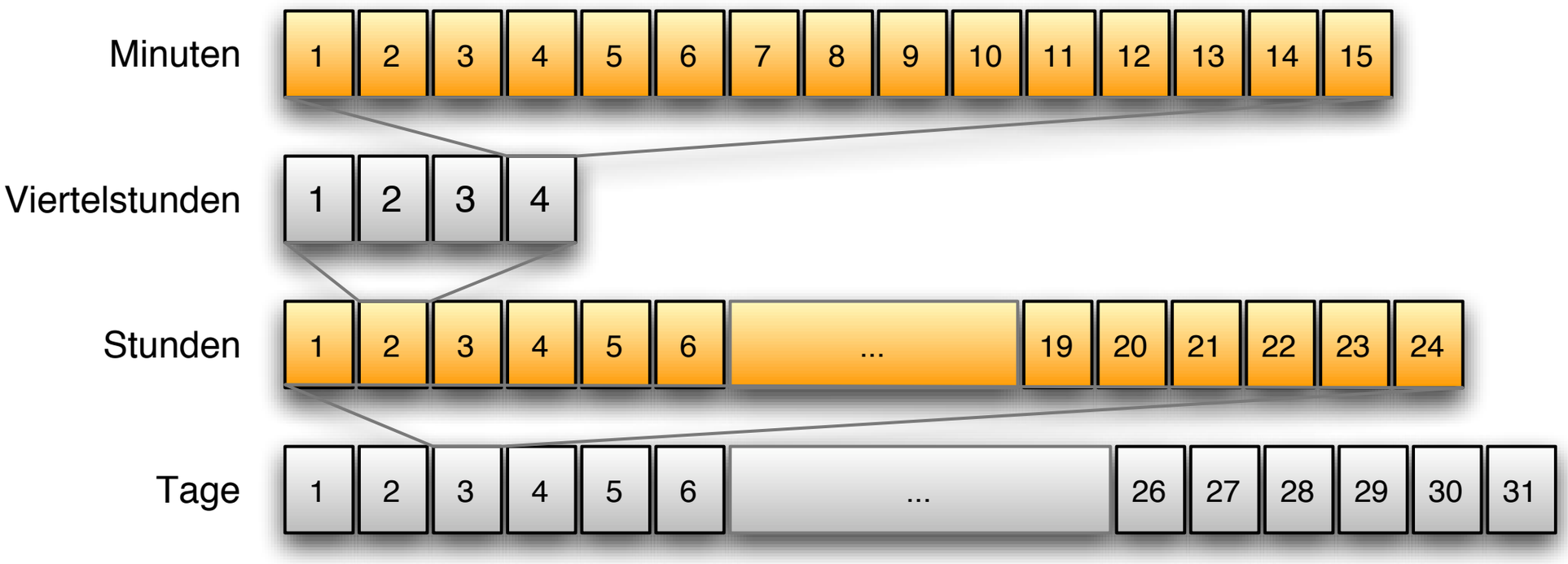


Geneigte Fenster

Um Daten aus einem möglichst langem Zeitraum zu speichern, verwendet MAIDS **geneigte Fenster:**



Datenstruktur:



Vergleich der Systeme

Kriterien	STREAM	Aurora	PIPES	SPEX
Einmalige Anfragen				
Kontinuierliche Anfragen				
Vordefinierte Anfragen				
Ad-hoc-Anfragen				
Blockierende Operatoren				
Zustandsbehaftete Operatoren				
Besonderheiten	**CQL**	**QoS**	**HMTS**	**XML**

Fazit

- Ansätze unterschiedlich, hohe Diversität
- Noch gibt es keinen klaren Sieger
- MAIDS: Die Killer-Applikation von Morgen?
- Datenströme mit Sicherheit zunehmend wichtig! Forschung ist daher von großer Bedeutung.

Dankeschön!

Fragen?