POZNAN UNIVERSITY OF TECHNOLOGY

On Analyzing Sequences and Building Sequential Data Warehouse

Robert Wrembel
Poznan University of Technology
Institute of Computing Science
Poznań, Poland
Robert.Wrembel@cs.put.poznan.pl
www.cs.put.poznan.pl/rwrembel

Outline

- **Introduction**
  - **ordered data and time-aware models**
- **Processing ordered data → overview**
  - **Time Series**
  - **Complex Event Processing**
  - **Sequences**
- **Analyzing sequences → overview**
  - **searching for patterns**
  - **OLAP on data streams**
  - **warehousing and OLAP**
- **Seq-SQL @PUT (Poznan University of Technology)**
  - **our approach to warehousing sequential data**

Ordered Data

- **Analysis of data items (observations, events, signals) whose order matters**
  - **typically, data items are ordered by time**
    - **scientific and engineering data**
    - **sensor measurements**
    - **power supply and consumption measurements**
    - **computer network traffic**
    - **stock exchange data**
    - **air pollution monitoring data**
    - **click stream**
    - **query logs**
- **Point-based events**
- **Interval-based events**

Point-based

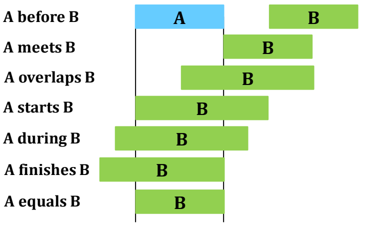
- **Event: <value, timestamp>**
  - **duration: instant or duration time is irrelevant**
- **Relations between events**
  - **before, after, equals**
- **Examples**
  - **stock exchange data**
  - **Web click stream**
  - **query logs**

Interval-based

- **Event: value, duration**
  - **duration: <TS_{beg}, TS_{end}>**
  - **duration: <TS_{beg}, time period>**
- **Support for temporal relations**
  - **starts-with, during, overlapping, within**
  - **temporal aggregation operators like**
    - **count started**
    - **count finished**
- **Relations between intervals → a few models**
  - F. Moerchen: Unsupervised pattern mining from symbolic temporal data. SIGKDD Explorations, (9)1, 2007

J. F. Allen. Maintaining knowledge about temporal intervals. CACM, 26(11), 1983

+ inverse relations



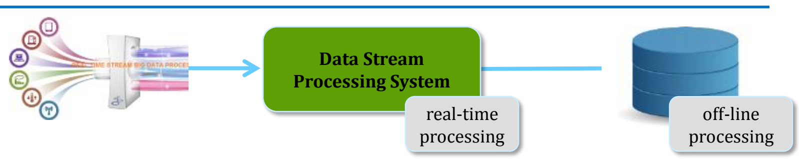
Coupling TB and IB Models

- **Intervals are shorthand for time points: conversion PB → IB (when the semantics of duration is not important)**
  - R. T. Snodgrass. The Temporal Query Language TQuel. ACM TODS, 12(2), 1987
  - A. Tansel, J. Clifford, S. Gadia, S. Jajodia, A. Segev, and R. T. Snodgrass. Temporal Databases: Theory, Design, and Implementation. Benjamin/Cummings, 1993
  - J. Chomicki. Temporal Query Languages: a Survey. Conf. on Temporal Logic, 1994
  - D. Toman. Point-based vs Interval-based Temporal Query Languages. PODS, 1996
  - N.A. Lorentzos, Y.G. Mitsopoulos: SQL Extension for Interval Data. TKDE, 9(3), 1997
- **Intervals have semantics**
  - M. H. Böhlen, R. Busatto and C. S. Jensen: Point- Versus Interval-based Temporal Data Models. ICDE, 1998

Temporal Databases

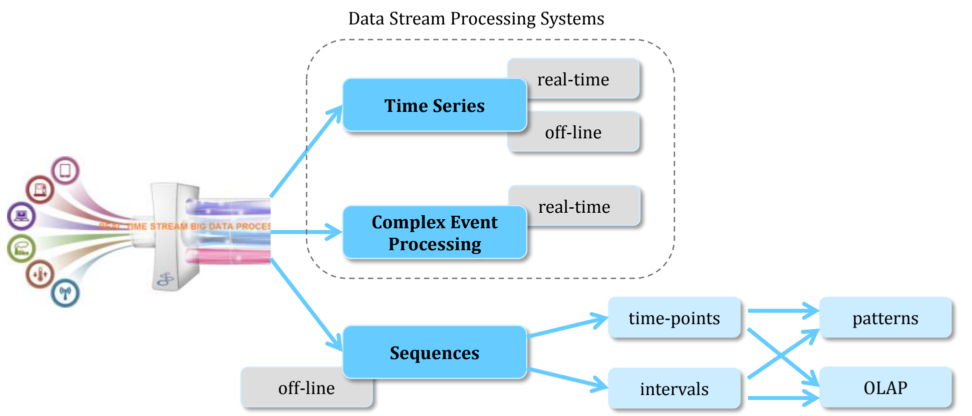
- **SQL-92**
  - **introduced interval data type**
- **TSQL2**
  - **temporal aggregates**
    - N. Kline, R.T. Snodgrass: Computing temporal aggregates. ICDE, 1995
  - **temporal algebra**
    - R.T. Snodgrass. The TSQL2 Temporal Query Language. Kluwer, 1995
- **Time interval-based query languages**
  - **IXSQL**
    - N.A. Lorentzos, Y.G. Mitsopoulos: SQL Extension for Interval Data. TKDE, 9(3), 1997
  - **ATSQL**
    - M. H. Böhlen, R. Busatto and C. S. Jensen: Point- Versus Interval-based Temporal Data Models. ICDE, 1998

Data Stream Processing Systems



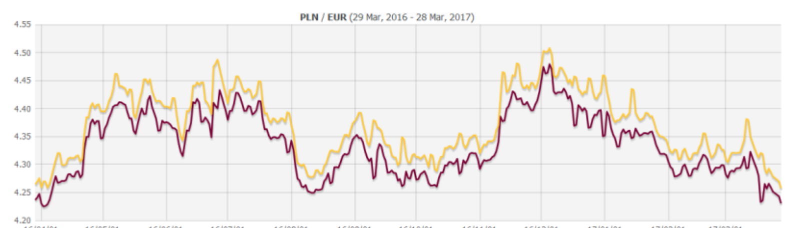
- **Ordered data as data stream**
- **DSPS: basic functionality**
  - **computing in real-time aggregates in a sliding window**
- **Systems (real-time processing)**
  - **Apache Storm**
  - **Apache Flink**
  - **Apache Kafka Streams**
  - **Apache Spark Streaming**
  - **Apache Samza**
  - **DataTorrent RTS**
  - **TIBCO StreamBase**
  - **...**

Ordered Data



Time Series

- A **time series** consists of values (elements, events) ordered by time
  - taken at successive **equally spaced points in time**
    - at a given frequency
  - variables of continuous values
- Examples
  - signals from sensors
  - financial
  - voice



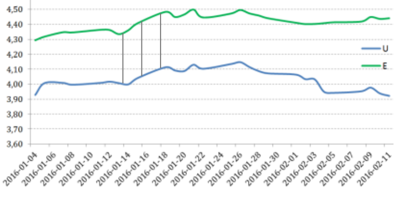
Time Series Analysis

- **Past is known → predicting the future**
- **Trend analysis**
- **Aggregating in a sliding window**
- **Detecting dangerous events / outliers**
- **Finding similarities between TS**
  - D. Rafiei, A.O. Mendelzon: Querying Time Series Data Based on Similarity, TKDE, 12(5), 2000
- **Pattern analysis**
  - **finding patterns in TS → sequential pattern mining on discrete sequences**
  - **searching for TS with a given pattern**
- **Classification & clustering → similarities**

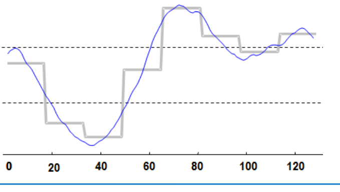
Time Series Analysis

- **Representations for similarity analysis**
  - **distance between two TS**

$$Dist(E,U) = \sqrt{\sum_{i=1}^{n}(e_i - u_i)^2}$$



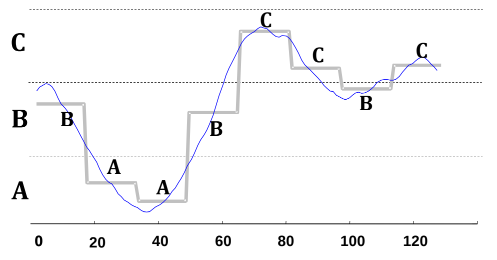
  - **Piecewise Aggregate Approximation (PAA): divide a TS into equal parts, represent each part by its AVG**



Time Series Analysis

- **Representations**
  - **Symbolic Aggregate approXimation (SAX)**
    - **uses Piecewise Aggregate Approximation**

J. Lin, E. Keogh, L. Wei, S. Lonardi: Experiencing SAX: a Novel Symbolic Representation of Time Series. Data Mining and Knowledge Discovery (15):2, 2007

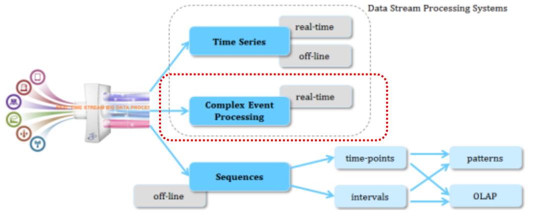


SAX representation: BAABCCBC

  - **Piecewise Linear Approximation (PLA)**
  - **Discrete Fourier Transform**
  - **...**

Complex Event Processing

Systems
- **SASE**
- **ZStream**
- **Cayuga**

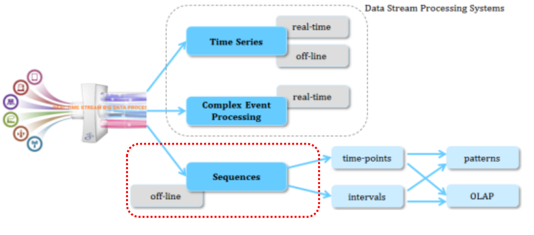


- **CEP engine**
  - **for processing large numbers of real-time events**
    - **e.g., trading, infrastructure monitoring, supply chain management, click-stream analysis, network intrusion detection, fraud detection**
  - **large number of concurrent queries on streams of events**
    - **detecting patterns and outliers**
  - **do not support multidimensional analysis**

Complex Event Processing

- **Functionality**
  - filtering
  - in-memory caching
  - aggregation over windows
  - database lookups
  - database writes
  - joins
  - queries (request-response, subscription)
  - producing hierarchical events
    - e.g., events from multiple sensors aggregated into events on a "hub" that integrates the sensors
  - advanced pattern matching (in real-time)
    - complex AND / OR expressions
    - negation

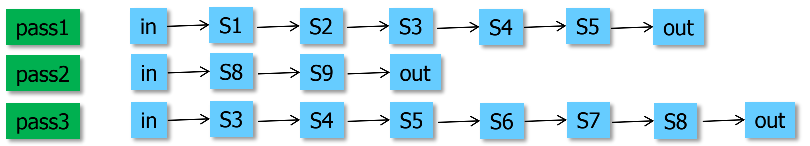
Sequences



- A sequence consists of ordered values (elements, events) recorded with or without a notion of time
  - numerical properties (quantify an event)
  - text properties (describe an event)
- Point-based sequences
- Interval-based → sequences of intervals

Sequences

- **Commuters' flow in a public transportation infrastructure**



the number of round-trips (e.g., S1 → S2 → S2 → S1) and their distributions over origin-destination within Q1 of 2017

- **Other examples**
  - **navigation between web pages**
  - **identification of pattern of purchases over time**
  - **sequence of search queries**
  - **alarm logs**
  - **workflow management systems**
  - **money laundry scenarios**
  - **...**

Sequences

- **Sequence analysis**
  - **offline → the whole sequence is available in advance**
  - **discovering unknown patterns → sequential pattern mining**
  - **prediction → Markov models**
  - **general purpose processing (searching for known patterns)**
  - **OLAP-like analysis (by means of SQL-like languages)**

Research Focus (sequences)

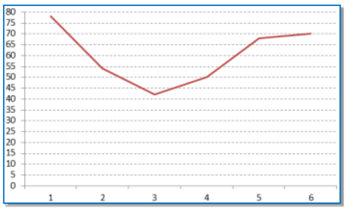
- **General purpose sequence processing**
  - SEQ: [SLR94,SLR95,SLR96]
  - SRQL: [RDR+98]
  - SQL-TS: [SZZA01a,SZZA01b,SZZA04]
  - AQuery: [LS03]
  - PartOrder: [MM00]
  - SciQL: [ZKM13]
  - Commercial: Oracle11g, 12, Aster
  - Open source: Hive, Spark
- **Features**
  - general purpose
  - off-line processing
  - point based

Research Focus (gen. purp. seq. proc.)

- **Features cont.**
  - SQL-like language
  - data models
    - relational, extended relational, array, multi-dimensional, partial order + probabilistic model
  - operations
    - selection
    - projection
    - join
    - offset access
    - basic aggregation functions: Count, Min, Max, Sum, Avg
    - pattern expression
  - basic query optimization idea
    - eliminate the set of sequences as soon as possible

Research Focus (gen. purp. seq. proc.)

- **Example**
  - **SQL-TS [SZZA01a,SZZA01b,SZZA04]**
  - **focuses on pattern search: provides a pattern search algorithm**




```
SELECT X.date AS start_date, X.price
       U.date AS end_date, U.price
FROM quote
CLUSTER BY name
SEQUENCE BY date
AS (X, Y, Z, T, U)
WHERE X.name='IBM'
   AND Y.price < X.price
   AND Z.price < Y.price
   AND 40 < Z.price < 50
   AND Z.price < T.price
   AND T.price < 52
   AND T.price < U.price
```

  - **Aster**

```
SELECT path, count(*)
FROM NPATH(ON tram
           PARTITION BY run_id ORDER BY time ASC
           MODE(OVERLAPPING)
           PATTERN('a.b.c.d.e')
           SYMBOLS (true as a, true as b, true as c, true as d, true as e)
           RESULT (ACCUMULATE(
                  station OF ANY(a,b,c,d,e)) AS path))
GROUP BY path
ORDER BY 2 DESC;
```


Research Focus (stream + OLAP)

- **Building cubes on streaming data for real-time OLAP-like analysis**
  - **Stream Cube [CDH+02,HCD+05]**
  - **E-Cube [LR10,LRR+10,LRG+10,LRG+11]**
- **Features**
  - **streaming data stored in data cubes**
  - **only parts of the cubes are materialized**
    - **the most frequently access paths in dimensions**
    - **algorithm for efficient cube computation**
      - based on hyperlink-tree (Stream Cube)
      - sharing query results (E-cube)
  - **no formal model**

Research Focus (stream + OLAP)

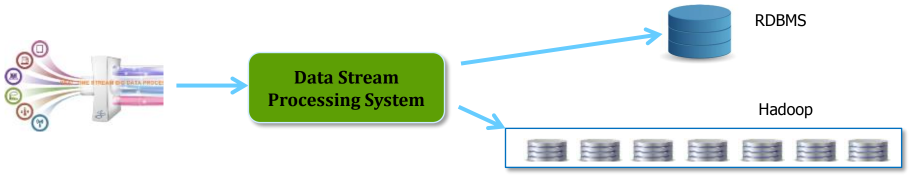
- **Features cont.**
  - **searching for complex patterns**
    - **with negation → E-Cube**
  - **multidimensional analysis of event patterns**
    - **a pattern dimension**
    - **pattern hierarchies**
  - **roll-up optimization**
- **Example (E-Cube)**

```
PATTERN <event pattern>
WHERE
GROUPBY
AGG
WITHIN <window>
```

```
PATTERN  SEQ(Recycle r, Washing w,
              ! SEQ(Sharpening s, Disinfection d, Checking c, s.id=d.id=c.id=o.id),
               Operating o, r.id=w.id=o.id and o.ins-type="surgery")
WITHIN   1 hour
```

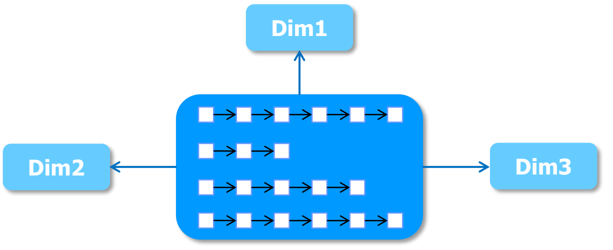

Research Focus (stream + OLAP)

- **Stream Data Warehouse**
  - Golab, et.al.: Data Stream Warehousing. Tutorial, ICDE, 2014
- **Process and collect streaming data from multiple sources**
- **Access real-time and historical data**
- **Technologies involved**
  - **RDBMS for some historical data and metadata**
  - **Hadoop for massive historical data**
  - **DSMS for streaming data**



Research Focus (sequences)

- Need for analyzing sequential data as cubes → sequential data warehouse
  - facts
  - dimensions
- Analysis
  - searching for known patterns
  - OLAP-like aggregations



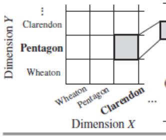
Research Focus (sequences as cubes)

- Grouping and aggregating by patterns
  - S-OLAP [LKH+08,Chu08,CLKH09,CKLC10, RL10,CKLC11, HWK+13,HWK+17]
  - application area: commuter flows
- Warehousing RFID data
  - FlowCube [GHL06a,GHLK06,GHL06b]
  - RFID DW [CKRS04]
  - application area: good transportation (RFID)
- Log analysis
  - Search logs [ZJPL09]
  - Process Cube [vdA13]
  - application area: analyzing log data
- Sequences of intervals
  - TIDAQL [MKM+15b,MKM+15a]
  - application area: general

S-OLAP

- **Sequences organized as cubes**
- **Dimensions may have concept hierarchies**
  - **station → district**
  - **time → day → month**
- **Pattern dimension**
- **Idea**
  - **group sequences having the same pattern in a cell**
  - **apply aggregate function (Count, Sum, Avg) to each group → result is s-cuboid**

Passenger (Sequence) ID	*Event Sequence*
...	...
s28	⟨ [t_4;Clarendon;enter;0], [t_7;Pentagon;exit;1.9], [t_9;Pentagon;enter;0],[t_{10};Clarendon;exit;1.9] ⟩
...	...
s623	⟨ [t_1;Pentagon;enter;0], [t_9;Wheaton;exit;1.9] ⟩
...	...



(X, Y, Y, X)	COUNT
(Clarendon, Pentagon, Pentagon, Clarendon)	16,289
(Clarendon, Wheaton, Wheaton, Clarendon)	2,654
...	...
(Rockville, TwinBrook, TwinBrook, Rockville)	5
...	...
(Pentagon, Wheaton, Wheaton, Pentagon)	6,543

Lo et.al.: OLAP on Sequence Data. SIGMOD, 2008

S-OLAP

- **Pattern based aggregate (PBA) queries**

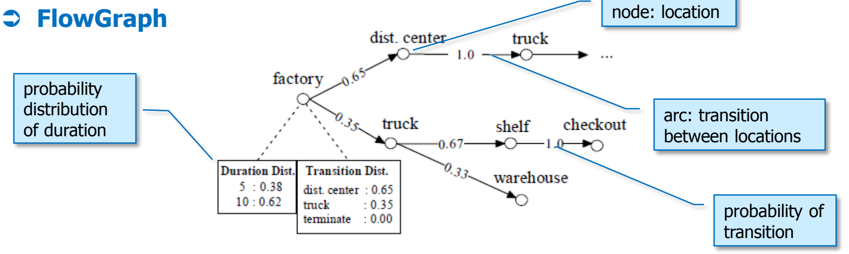
```
SELECT              COUNT(*)
FROM                Event
WHERE               time >= 2007-10-01T00:00 AND
                    time < 2007-12-31T24:00
CLUSTER BY          card-id AT individual,
                    time AT day
SEQUENCE BY         time ASCENDING
SEQUENCE GROUP BY   card-id AT fare-group,
                    time AT day
CUBOID BY           SUBSTRING (X, Y, Y, X) WITH
                    X AS location AT station,
                    Y AS location AT station
                    LEFT-MAXIMALITY (x1, y1, y2, x2) WITH
                    x1.action = "in" AND
                    y1.action = "out" AND
                    y2.action = "in" AND
                    x2.action = "out"
```

S-OLAP

- **Cube computation**
  - **partially materialize** **the cube**
  - **inverted list** **to support finding sequences including a given pattern**
    - **pattern ➔ {sequence ID, {positions within the sequence where a given pattern begins}}**
  - **method for building complex patterns from simpler ones**
    - **inverted list joining and computing values of cells**
- **Iceberg cells computation**
  - **synopsis-based algorithm (SBA)**
    - **synopsis** **of sequences maintained in RAM**
    - **synopsis is based on uniform** **sampling** **of sequences**
    - **use** **statistical tests on the synopsis to decide whether a cell is iceberg** **at a given significance level**
    - **for the identified potential iceberg cells,** **estimate** **their** **aggregate** **values based** **on the synopsis**

FlowCube

- **For analyzing paths of goods transportaion**
- **FlowGraph**



- **FlowCube**
  - **a cell stores a FlowGraph**
  - **cells are referenced by hierarchical dimensions (e.g., Location)**
  - **roll-up, drill-down along a dimension hierarchy**
  - **pattern mining on cube cells**
- **No query language**
- **No data model**

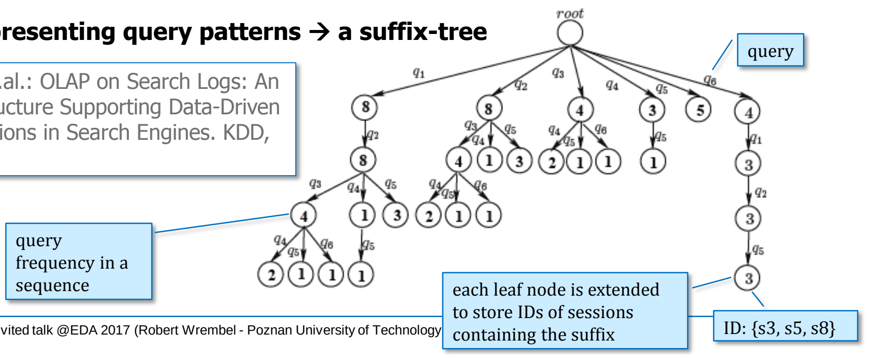
Gonzalez et.al.: FlowCube: Constructing RFID FlowCubes for Multi-Dimensional Analysis of Commodity Flows. VLDB, 2006

OLAP on Search Logs

- **Idea**
  - **each query in a query sequence S is a dimension**
  - **the frequency of a query is a measure**
  - **sequential drill-down on S**
    - **prepending sequence S1 at the head of S**
    - **appending S1 at the tail of S**
  - **sequential roll-up on S**
    - **removing a subsequence of S either at the head or tail**

Representing query patterns → a suffix-tree

Zhou et.al.: OLAP on Search Logs: An Infrastructure Supporting Data-Driven Applications in Search Engines. KDD, 2009



Process Cube

- **To organize workflows as cube cells for process mining**
- **Components**
  - **event database (events have properties that have values)**
  - **the structure of a cube (dimensions, hierarchies) → Process Cube**
    - **a cube cell stores process instances**
  - **view (select dim. instances, slice, dice) → Process Cube View**
  - **materialized view (instantiation of PCV) → Materialized Process Cube View**
- **Operations provided:**
  - **slice/dice**
  - **roll-up/drill-down (no discussion how to represent workflows in cube cells)**
- **No query language**

Research Focus (sequences)

➲ Indexing

- **various alternatives of suffix trees, e.g.,**
  - Zhou et.al.: OLAP on Search Logs: An Infrastructure Supporting Data-Driven Applications in Search Engines. KDD, 2009
  - Andrzejewski et.al.: FOCUS: An Index for Continuous Subsequence Pattern Queries. ADBIS, 2012
- **inverted index, e.g.,**
  - Lo et.al.: OLAP on Sequence Data. SIGMOD, 2008
  - Zhou et.al.: OLAP on Search Logs: An Infrastructure Supporting Data-Driven Applications in Search Engines. KDD, 2009

Summary: Approaches

Contribution	DModel	Storage	QL	QOpt	Indx	L.l.oper.	H.l.oper.	Dom.	Cube sup.	PSearch	TPoint	Interv.	Applic.
SEQ: [SLR94,SLR95,SLR96]	ext.rel.	rel.	yes	yes	no	no	yes	general	no	no	yes	no	OLTP
SRQL: [RDR+98]	ext.rel.	rel.	yes	no	no	yes	no	general	no	no	yes	no	OLTP
SQL-TS: [SZZA01a,SZZA01b,SZZA04]	rel.	rel.	yes	yes	no	no	yes	general	no	yes	yes	no	OLTP
AQuery: [LS03]	arrable	ext. rel., col.	yes	yes	no	yes	?	general	no	no	yes	no	OLTP
PartOrder: [MM00]	part.ord.+prob.	no	no	no	no	no	no	general	no	no	yes	no	OLTP
SciQL: [ZKM13]	array	array	yes	no	no	?	?	general	no	no	yes	no	OLTP
StreamCube: [CDH+02,HCD+05]	rel.	MD	no	yes	no	no	no	stream	yes	no	yes	no	OLAP
e-Cube: [LR10,LRR+10,LRG+10,LRG+11]	rel.	stream	yes	yes	no	no	yes	stream	yes	yes	yes	no	OLAP
S-OLAP: [LKH+08,Chu08,CLKH09,CKLC10, RL10,CKLC11,HWK+13,HWK+17]	rel.	rel.	yes	yes	inv.indx	no	yes	com.flows	yes	SQL ext.	yes	no	OLAP
FlowCube: [GHL06a,GHLK06,GHL06b]	no	MD	no	no	no	no	no	RFID	yes	no	yes	no	OLAP
RFID DW: [CKRS04]	no	rel.	no	no	no	no	no	RFID	yes	no	yes	no	OLAP
Log OLAP: [ZJPL09]	?	MD	no	yes	yes	no	yes	logs	yes	yes	yes	no	OLAP
ProcessCube: [vdA13]	dedicated	MD	no	no	no	no	yes	workflows	yes	DM alg.	?	no	OLAP
TIDAQL: [MKM+15b,MKM+15a]	no	rel.	yes	yes	yes	no	yes	general	yes	no	no	yes	OLAP
I-OLAP: [KMWE14]	dedicated	rel.	no	no	no	yes	yes	general	yes	no	no	yes	OLAP
SEQ-SQL: [BMM+12,BMKW14,KPW15, BCM+15,BKW16]	dedicated	rel.	yes	no	no	yes	yes	general	yes	yes	yes	no	OLAP
Oracle12c	rel.	rel.	yes	yes	yes	no	yes	general	no	M_R	yes	no	OLTP
Aster	rel.	rel.	yes	yes	?	no	yes	general	no	nPath	yes	no	OLTP
Hive	NoSQL	NoSQL	yes	?	?	no	yes	general	no	MatchPath	yes	no	OLTP
Spark	NoSQL	mem.	yes	?	?	no	yes	general	no	MatchPath	yes	no	OLTP

Seq-SQL @PUT

➲ Example sequential data

	failure date	car ID	model	make	prod. year	mileage	failureID	rep. cost
e1	2012.04.04	BB111	308	Peugeot	2003	145 500	F1	1 500
e2	2012.06.11	BB111	308	Peugeot	2003	160 000	F2	800
e3	2012.06.12	AA222	207	Peugeot	2004	184 000	F3	2 100
e4	2012.06.13	CC333	Golf	VW	2007	80 000	F2	790
e5	2013.07.27	BB111	308	Peugeot	2003	179 000	F3	2 200
e6	2012.12.02	AA222	207	Peugeot	2004	201 123	F4	650
e7	2012.12.08	CC333	Golf	VW	2007	120 000	F4	660
e8	2012.12.13	DD444	Polo	VW	2005	110 000	F1	1 400
e9	2013.01.30	EE555	Octavia	Skoda	2000	190 000	F3	1 900
e10	2013.02.16	DD444	Polo	VW	2005	121 000	F2	850

➲ Data model elements

- **event**
- **sequence**
- **measure**
- **dimension**
- **dimension hierarchy**

Seq-SQL @PUT

➲ Event

- **elementary data item that lasts for an instant**
- **events have attributes**
  - **descriptors**
  - **measures**
  - **dimensions**
    - an attribute may form a hierarchy
      - failure date → month → quarter → year
      - carID → model → make → manufacturing company
      - failureID → failureType

	failure date	car ID	model	make	prod. year	mileage	failureID	rep. cost
e1	2012.04.04	BB111	308	Peugeot	2003	145 500	F1	1 500
e2	2012.06.11	BB111	308	Peugeot	2003	160 000	F2	800
e3	2012.06.12	AA222	207	Peugeot	2004	184 000	F3	2 100
e4	2012.06.13	CC333	Golf	VW	2007	80 000	F2	790
e5	2013.07.27	BB111	308	Peugeot	2003	179 000	F3	2 200
e6	2012.12.02	AA222	207	Peugeot	2004	201 123	F4	650
e7	2012.12.08	CC333	Golf	VW	2007	120 000	F4	660
e8	2012.12.13	DD444	Polo	VW	2005	110 000	F1	1 400
e9	2013.01.30	EE555	Octavia	Skoda	2000	190 000	F3	1 900
e10	2013.02.16	DD444	Polo	VW	2005	121 000	F2	850

Seq-SQL

➲ Sequence

- a list of events
- created by operator CreateSequence
  - input: the set of events
  - output: a set of sequences
  - selects events that fulfill a given condition
  - a sequence is created from events having the same value of a **forming attribute**
  - events within a sequence are ordered by an **ordering attribute** (typically timestamp)

Seq-SQL

- CreateSequence(failures, car_id, car_mileage, null)

```
create seq failures_seq on failures
forming attributes car_id
ordering attributes car_mileage
```

s1	e1	2012.04.04	BB111	308	Peugeot	2003	145 500	F1	1 500
	e2	2012.06.11	BB111	308	Peugeot	2003	160 000	F2	800
	e5	2013.07.27	BB111	308	Peugeot	2003	179 000	F3	2 200
s2	e3	2012.06.12	AA222	207	Peugeot	2004	184 000	F3	2 100
	e6	2012.12.02	AA222	207	Peugeot	2004	201 123	F4	650
s3	e4	2012.06.13	CC333	Golf	VW	2007	80 000	F2	790
	e7	2012.12.08	CC333	Golf	VW	2007	120 000	F4	660
s4	e8	2012.12.13	DD444	Polo	VW	2005	110 000	F1	1 400
	e10	2013.02.16	DD444	Polo	VW	2005	121 000	F2	850
s5	e9	2013.01.30	EE555	Octavia	Skoda	2000	190 000	F3	1 900

Seq-SQL

- **CreateSequence**
  (failures, car_id, car_mileage, mileage>130000)

```
create seq failures_seq on failures
forming attributes car_id
ordering attributes car_mileage
where mileage>130000
```

s1	e1	2012.04.04	BB111	308	Peugeot	2003	145 500	F1	1 500
	e2	2012.06.11	BB111	308	Peugeot	2003	160 000	F2	800
	e5	2013.07.27	BB111	308	Peugeot	2003	179 000	F3	2 200
s2	e3	2012.06.12	AA222	207	Peugeot	2004	184 000	F3	2 100
	e6	2012.12.02	AA222	207	Peugeot	2004	201 123	F4	650
s5	e9	2013.01.30	EE555	Octavia	Skoda	2000	190 000	F3	1 900

Seq-SQL

➲ **Operations on sequences: transform a set of sequences into another set of sequences**
  - **First, Last**
  - **Subsequence**
  - **Split, Combine**
  - **Select sequences**
  - **Select events**
  - **Join**
  - **Union, Difference, Intersection**
  - **Drill-down, Roll-up**
  - **Group by**
  - **Aggregate**

Seq-SQL

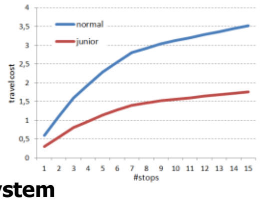
➲ **Split: divides sequences into the set of new sequences based on values of a given expression**

s1	e1	2012.04.04	BB111	308	Peugeot	2003	145 500	F1	1 500
	e2	2012.06.11	BB111	308	Peugeot	2003	160 000	F2	800
	e5	2013.07.27	BB111	308	Peugeot	2003	179 000	F3	2 200

```
select seq ev.dimension.dim_time.day, failure_description
from failures_seq
where event [ev.dimension.dim_time.year > 2012]
split on e_d_dim_time_year
```

s11	e1	2012.04.04	BB111	308	Peugeot	2003	145 500	F1	1 500
	e2	2012.06.11	BB111	308	Peugeot	2003	160 000	F2	800
s12	e5	2013.07.27	BB111	308	Peugeot	2003	179 000	F3	2 200

Seq-SQL



➲ **Fact = sequence**

➲ **Measure**

- **a property of an event**
- **a property of the whole sequence**
  - **the cost of a trip in a public transportation system**
  - **sequence length**
- **created by operator ComputeMeasure**
  - **input: a set of sequences**
  - **output: measure**
  - **ComputeMeasure(S, total_cost, sum(sequence.repair_cost))**

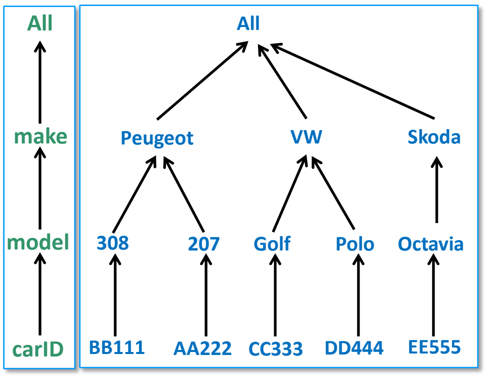
```
create seq measure total_cost
on failures_seq as
  [select sum(sequence.repair_cost)
   from dual]
   type [number(10)]
```

Seq-SQL

- **Dimension**
  - defines a context for analysis
  - an attribute of an event or
  - a property of the whole sequence
    - trip segment (short: 1-4 stops, medium: 5-9 stops, long: >9 stops)
  - may be hierarchical
  - defined by operator CreateContext
    - name
    - event attribute | compute dimension
    - optional set of dimension hierarchies

Seq-SQL

```
create seq dimension car_dim
on failures_seq
level car as
  [select distinct(f.car_id)
   from failures f, failures_seq s
     where s.event=f.rowid
       and s.seq_id=sequence.seq_id
  ] type [varchar2(20)]
is child of level model as
  [select distinct car_model
   from failures
   where car_id = event.value
  ] type [varchar2(100)]
is child of level make as
  [select distinct car_make
   from failures
   where car_model = event.value
  ] type [varchar2(100)]
```



Seq-SQL

➲ Example analysis

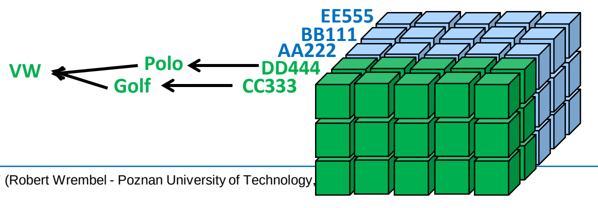
for every model of make VW count cases of failures where the same failure occurred within mileage < 10000 km and at least one other failure occurred in between, consider only repairs of the total cost >= 5000

```
select seq seq.dimension.car_dim.model,
       [count(distinct seq.seq_id) as num_cases]
from failures_seq
where seq [seq.measure.total_cost >= 5000
       and seq.dimension.dim_car.make = 'VW']
where pattern [XYX]
       with mapping [X to failure_description,
                     Y to failure_description]
       with options [X.2.car_mileage - X.1.car_mileage < 10000]
group by seq.dimension.car_dim.model
```

sequecne selection

slice

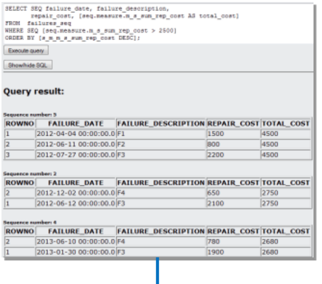
pattern



Prototype sequence DW (v.0)

➲ Code complexity in a standard approach

- 3 materialized views + complex SQL
- query (approx. 60 LOC) + Java code for
- analyzing patterns (approx. 300 LOC)



```
SELECT SEQ failure_date, failure_description,
       repair_cost, [seq.measure.m_s_sum_rep_cost AS total_cost]
FROM  failures_seq
WHERE SEQ [seq.measure.m_s_sum_rep_cost > 2500]
ORDER BY [s_m_m_s_sum_rep_cost DESC];
```

Execute query

Show/hide SQL

Query result:

Sequence number: 3

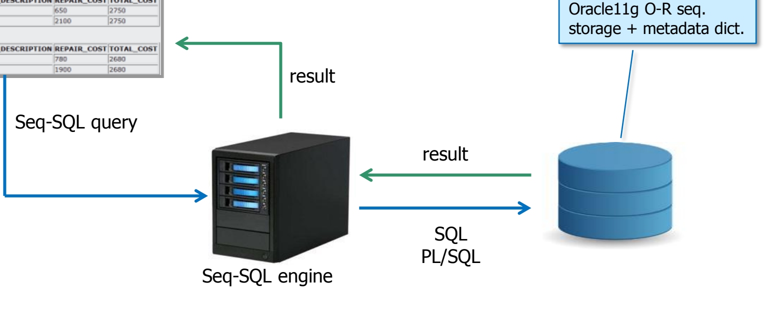
ROWNO	FAILURE_DATE	FAILURE_DESCRIPTION	REPAIR_COST	TOTAL_COST
1	2012-04-04 00:00:00.0	F1	1500	4500
2	2012-06-11 00:00:00.0	F2	800	4500
3	2012-07-27 00:00:00.0	F3	2200	4500

Sequence number: 2

ROWNO	FAILURE_DATE	FAILURE_DESCRIPTION	REPAIR_COST	TOTAL_COST
2	2012-12-02 00:00:00.0	F4	650	2750
1	2012-06-12 00:00:00.0	F3	2100	2750

Sequence number: 4

ROWNO	FAILURE_DATE	FAILURE_DESCRIPTION	REPAIR_COST	TOTAL_COST
2	2013-06-10 00:00:00.0	F4	780	2680
1	2013-01-30 00:00:00.0	F3	1900	2680



Seq-SQL ↔ ProcessCube

- **ProcessCube**
  - the lowest levels of dimensions defined on events attributes
- **Seq-SQL**
  - a dimension may be either the property of an event or the whole sequence
- **ProcessCube**
  - the notion of a measure is undefined
- **Seq-SQL**
  - a measure may be the property of either an event or the whole sequence
- **ProcessCube**
  - only high level operations on the process cube are defined (slice, dice, roll-up, drill-down)
- **Seq-SQL**
  - defines the whole formal model for sequential data
    - including 12 low level operations on sequences
    - based on these operations, higher level operations may be built

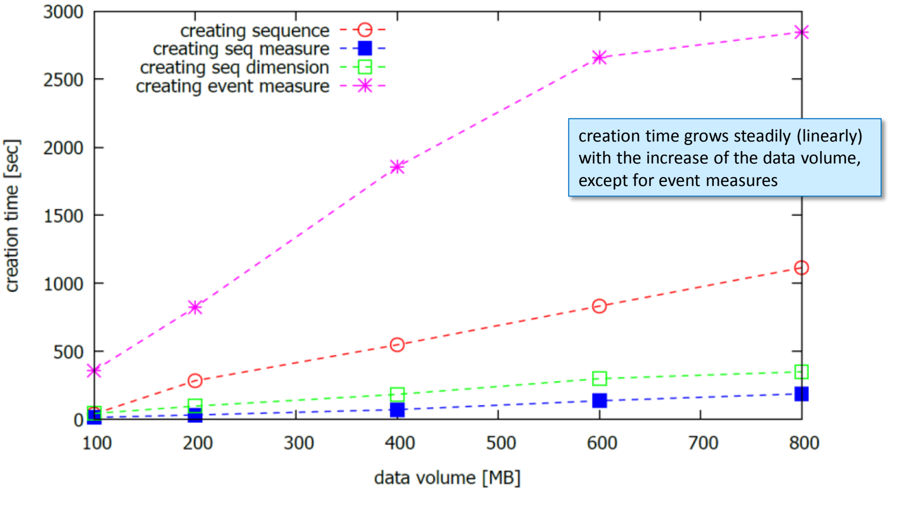
Seq-SQL ↔ S-OLAP

- **S-OLAP**
  - a dimension can be a property of either an event attribute or a pattern
- **Seq-SQL**
  - the definition of a dimension is more general → can be a property of an event or a property of the whole sequence, thus a pattern as well
- **S-OLAP**
  - no formal data model for sequences
- **Seq-SQL**
  - formal model defined
- **S-OLAP**
  - defines 6 operations that process patterns only
- **Seq-SQL**
  - provides 12 low level operations on sequences, measures, and dimensions

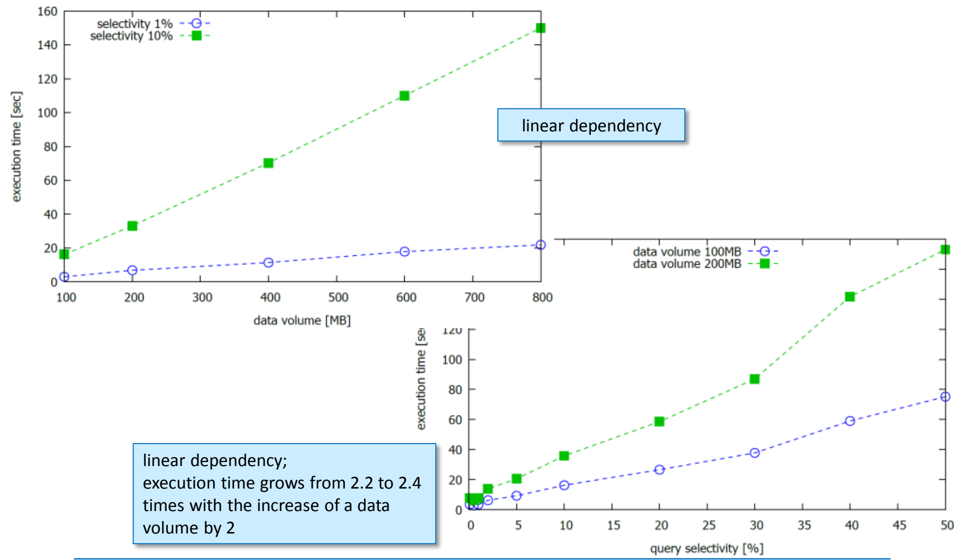
Experimental Evaluation

- **Data**
  - **sensor measurements**
  - **volumes: 100MB, 200MB, 400MB, 600MB, and 800MB → 10M, 20M, 40M, 60M, and 80M of events**
- **Measurements**
  - **creation time of DW objects: sequences, sequence measures, sequence dimensions, and event measures**
  - **exec. time of queries of multiple selectivities on multiple data volumes**
  - **...**

Creating objects



Selecting sequences



Storing Sequences

- Storage may impact performance
- Mutliple ways of storing sequences
  - relational table
  - noSQL "table"
  - graph
  - list
  - array
  - ...
- Different systems & syntax for patterns
  - Oracle Database 12c: Match_Recognize
  - Teradata Aster 6.00.01: NPath
  - Apache Hive 0.13.0: NPath
    - syntax similar to Aster; available in Hive v. 0.11.0, in v. 0.12.0 renamed to MATCHPATH, unavailable from v. 0.14.0
  - Apache Spark 1.3.0 (includes Hive)

goal: to evaluate

Evaluation: Query

- n-elements pattern query
  - includes exactly n elements in its pattern
  - example (n=5): find travels of passengers (CLUSTER BY cardID) that
    - started at tram stop 's14' (A.event type = 'in')
    - went through stop 's15'
    - went through 2 other stops
    - finished at the 5-th stop (E.event type = 'out')

SQL-TS

```
SELECT A.stopID, B.stopID, C.stopID, D.stopID, E.stopID
FROM travels
CLUSTER BY cardID
SEQUENCE BY timestamp AS (A, B, C, D, E)
WHERE A.stopID = 's14'
  AND B.stopID = 's15'
  AND A.event type = 'in'
  AND E.event type = 'out'
```


Evaluation: Query

- At least n-elements pattern query
  - includes at least n pattern elements in its AS clause
- Example (n>=3)

must appear once

must appear at least once

SQL-TS

```
SELECT A.stopID, B.stopID, C.stopID
FROM travels
CLUSTER BY cardID
SEQUENCE BY timestamp AS (A, *B, C)
WHERE C.cost - A.cost > 3
  AND C.cost - A.cost < 10
  AND A.event type = 'in'
```

must appear once

additional conditions

Evaluation: Data

- Artificially generated sequential data
  - representing passenger travels in the public transportation network of the city of Poznań, with 120 tram stops
  - poisson distribution of travel lengths (median=7.5) and card types
  - the structure of data generated by card readers → table
  - AVG row size = 30B

cardID (NUMBER)	a passenger's intelligent card ID
stopID (STRING)	a tram stop at which a given event occurred {'s1', ..., 's120'})
tramNo (STRING)	tram number
timestamp (DATETIME)	a timestamp of an event
card type (NUMBER)	passenger's card type {0, 1, 2, 3}
cost (NUMBER)	a cumulative cost of a travel
event type (STRING)	a type of an event: 'in', 'out', null

Evaluation: Setup

- The number of data items stored in the database
  - 24,055,341 (1 GB)
  - 121,020,658 (5GB)
  - 240,543,617 (10GB)
  - 480,057,689 (20 GB)
- The experiments conducted on 2 workstations
  - Intel R CoreTM i7-4700MQ
  - 16GB DDR3 1600MHz CL9 RAM, HDD Seagate ST1000LM014-1EJ164
  - Linux Ubuntu 14.10

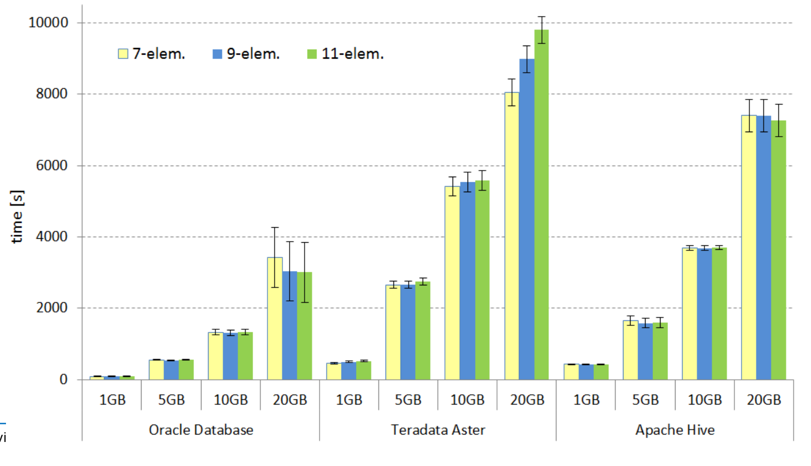
Evaluation: Queries

- **Find sequences of 3, 5, 7, 9, and 11 elements**
- **An example Oracle query finding 5-elements sequences**

```
SELECT stA, stB, stC, stD, stE, count(*)
FROM
   (SELECT * FROM travels
    MATCH_RECOGNIZE
       (PARTITION BY cardID ORDER BY timestamp ASC
        MEASURES a.stopID as stA, b.stopID as stB,
                 c.stopID as stC,
                 d.stopID as stD, e.stopID as stE
        ONE ROW PER MATCH AFTER MATCH SKIP TO NEXT ROW
        PATTERN(a b c d e)
        DEFINE a as a.event type = 'in', e as e.event type = 'out'))
GROUP BY stA, stB, stC, stD, stE
ORDER BY 6 DESC
```

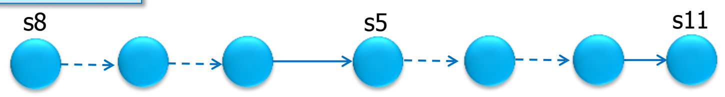

Evaluation: Queries

- **Aster - query execution times linearly depend on a sequence length and on a data volume**
- **Oracle and Hive - query execution times linearly depend on a data volume and they do not depend much on a sequence length**

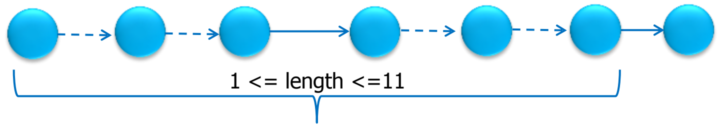


Evaluation: Queries

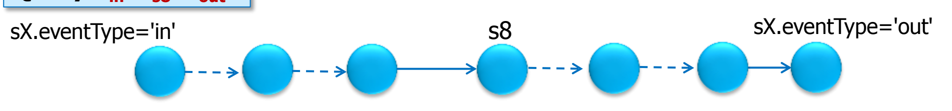
Query $A_{s8}*B_{s5}*C_{s11}$



Query $A*B_{length \leq 11}$

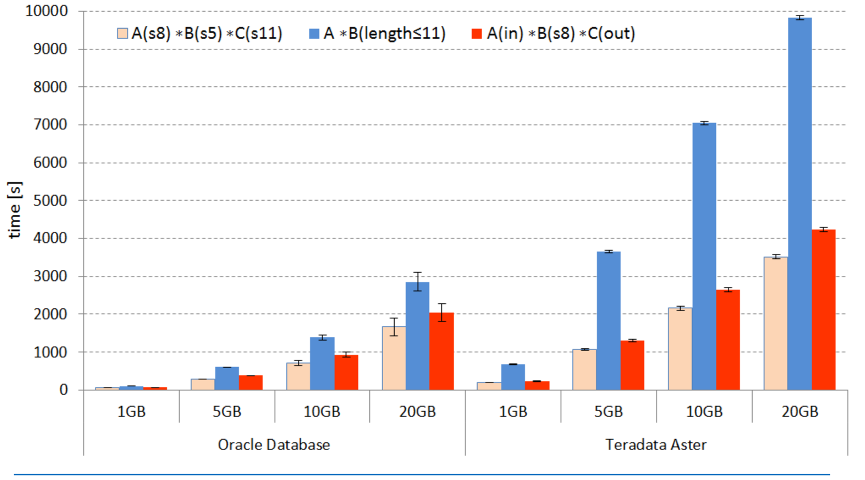


Query $A_{in}*B_{s8}*C_{out}$



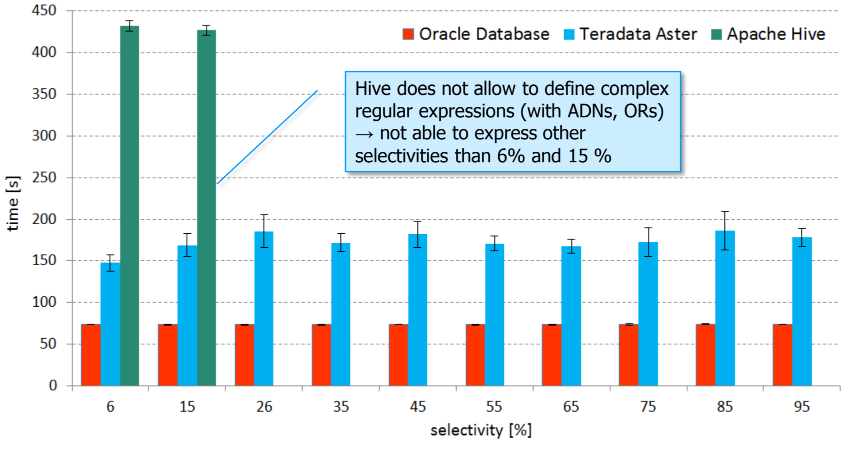
Evaluation: Queries

- **Hive does not allow to define queries with at least n-elements patterns**

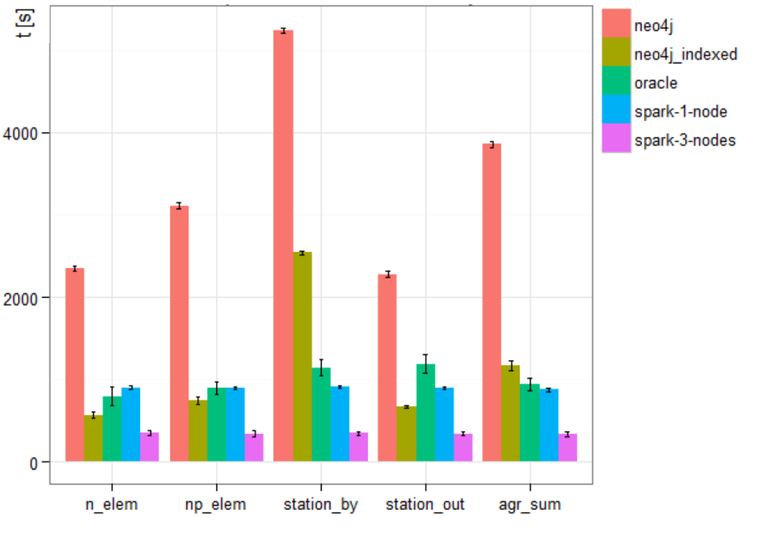


Evaluation: Queries

- ➲ **Impact of a query selectivity on the query execution time**
- ➲ **Query selectivity is defined as RetSeq/AllSeq**
  - **RetSeq is the number of sequences returned by a test query**
  - **AllSeq is the total number of sequences returned by a reference query (returning at least 2-stop trips)**

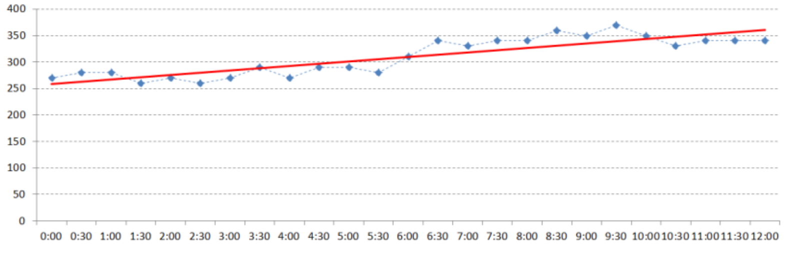


Evaluation



Seq-SQL: Intervals as Functions

- Idea: replace raw data within an interval by a linear regression model



- Pros
  - compression
  - approximation of a value in any time-point

Seq-SQL: Intervals as Functions

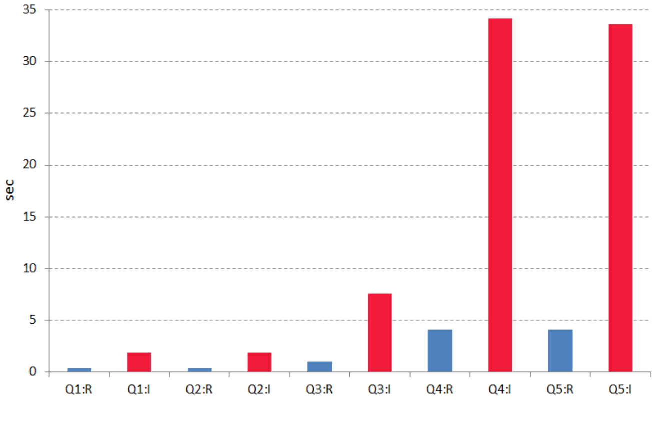
- Related approaches
- Chen et.al.: Multi-dimensional regression analysis of time-series data streams. VLDB, 2002
  - linear regression applied to time series
- Thiagarajan et.al.: Querying continuous functions in a database system. SIGMOD, 2008
  - functions: first class citizens, used in queries
  - function views: relational interface to access results of function executions
- Perera et.al.: Modeling large time series for efficient approximate query processing. DASFAA, 2015
  - raw data are used to create models
  - models used for generating approximate query answers (faster than querying raw data)

Seq-SQL: Intervals as Functions

- **Experimental setup**
  - **data: 1 reading per 30 min within 4-years period**
  - **storage: RDBMS**
  - **querying: SQL + stored functions**
  - **queries**
    - **Q1: for each meter, find its monthly energy usage in [2012 - 2015]**
    - **Q2: for each building, find monthly energy usage in [2012 - 2015] (hierarchy: meter → building)**
    - **Q3: select meter with MAX AVG energy usage during a day [8:15AM - 4:15PM] in the 2nd quarter of 2013**
    - **Q4: for each meter, find a working day in 2013 with MAX energy usage**
    - **Q5: for each meter, find working days in 2013 with energy usage > AVG of this meter**

Seq-SQL: Intervals as Functions

- **Results**
  - **R: raw relational data**
  - **I: interval representation**



Seq-SQL: Summary

- **Limitations**
  - **does not support mining of either patterns or processes**
  - **supports off-line analysis of sequences**
  - **not so advanced pattern matching as in S-OLAP**
- **Publications**
  - [BMM+12]
  - [BMKW14]
  - [KMWE14]
  - [KPW15]
  - [BCM+15]
  - [BKW16]
  - [ABK+16]
- **Open problems**
  - **to be handled by the National Science Center grant**

Projects@PUT

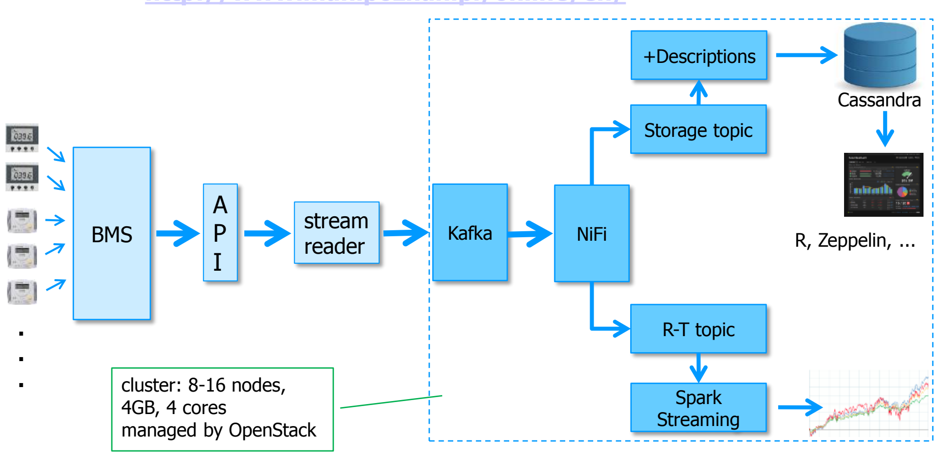
- **Analytical processing and mining of sequential data: models, algorithms, and data structures**
  - **National Science Center (NSC) grant No. 2015/19/B/ST6/02637**
  - https://www.ncn.gov.pl/?language=en
  - **one PhD scholarship is open**
- **Goals**
  - **G1: developing data models for representing time point-based and interval-based sequential data**
    - **low level elementary operations + their execution costs**
    - **query trees built with these operations → query optimization**
  - **G2: developing a fully functional query language for sequential data processing in an OLAP-like style**
    - **including complex pattern matching**

Projects@PUT

- **Analytical processing and mining of sequential data: models, algorithms, and data structures @NSC**
- **Goals**
  - **G3: developing data storage architectures suitable for OLAP processing of sequential data and their performance evaluation**
  - **G4: developing indexing techniques for sequential data and their performance evaluation**
  - **G5: developing mining algorithms for sequential data**
  - **G6: building a prototype architecture of sequential data warehouse as a service**

Projects@PUT

- **Analyzing monitoring signals from a data center**
  - **Poznan Supercomputing and Networking Center (PSNC)**
  - http://www.man.poznan.pl/online/en/



Projects@PUT

- Analysis of monitoring signals from data center @PSNC
- Processing
  - time series analysis
  - alarm signals → sequences → patterns
- Over 4400 different variables (signals) generated by BMS

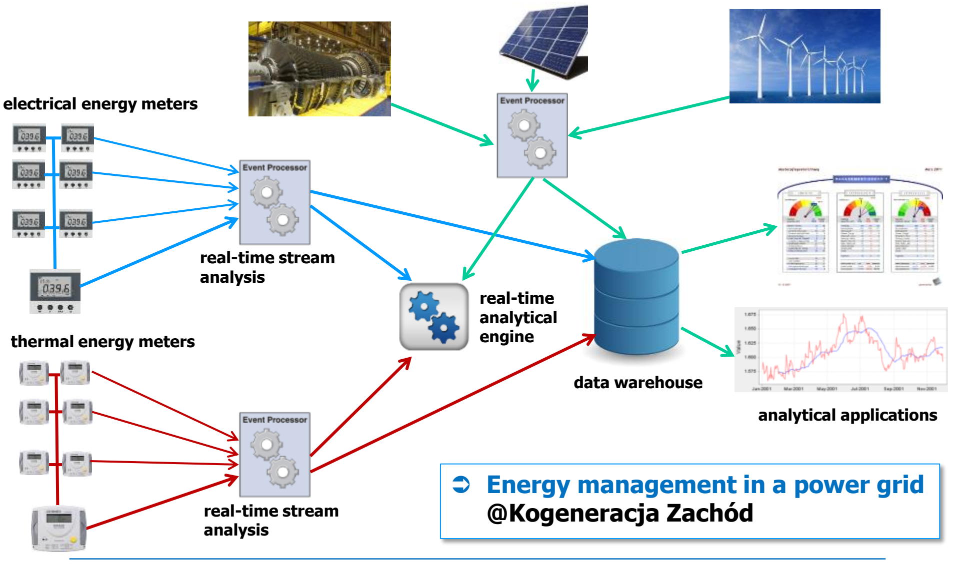
Projects@PUT

- Energy management in a power grid
  - Kogeneracja Zachód
  - http://kogeneracjazachod.pl/
  - one PhD scholarship envisaged → financed from project: Erasmus Mundus Joint Doctorate on Information Technologies for Business Intelligence - Doctoral College (IT4BI-DC)

Erasmus+

IT4BI DC
Information Technologies for Business Intelligence
Doctoral College

Projects@PUT



➲ **Energy management in a power grid @Kogeneracja Zachód**

Summary

➲ **Overview of techniques for processing ordered data**
- **time series**
- **complex event processing**
- **sequences**

➲ **The state of the art in processing and warehousing sequential data**

➲ **Seq-SQL @PUT**
- **SeqDW model**
- **implementation**
- **evaluation**
- **projects**

➲ **Open issues → NSC grant**

Rerferences

- [ABK+16] W. Andrzejewski, B. Bębel, S. Kłosowski, B. Łukaszewski, R. Wrembel, G. Bakkalian: Searching for Patterns in Sequential Data: Functionality and Performance Assessment of Commercial and Open-Source Systems. ER Workshops 2016.
- [BCM+15] B. Bębel, T. Cichowicz, T. Morzy, F. Rytwiński, R. Wrembel, and C. Koncilia. Sequential data analytics by means of seq-sql language. In DEXA, 2015.
- [BKW16] G. Bakkalian, C. Koncilia, and R. Wrembel. On representing interval measures by means of functions. In MEDI, 2016.
- [BMKW14] B. Bębel, T. Morzy, Z. Królikowski, and R. Wrembel. Formal model of time point-based
- sequential data for OLAP-like analysis. Bulletin of the Polish Academy of Sciences (Technical Sciences), 62(2), 2014.
- [BMM+12] B. Bębel, M. Morzy, T. Morzy, Z. Królikowski, and R. Wrembel. OLAP-like analysis of time point-based sequential data. In ER Workshops, 2012.
- [CDH+02] Y. Chen, G. Dong, J. Han, B.W. Wah, and J. Wang. Multi-dimensional regression analysis of time-series data streams. In VLDB, 2002.
- [Chu08] C.K. Chui. The design and implementation of an olap system for sequence data analysis. In IDAR@SIGMOD, 2008.
- [CKLC10] C.K. Chui, B. Kao, E. Lo, and D. Cheung. S-OLAP: an OLAP system for analyzing sequence data. In SIGMOD, 2010.
- [CKLC11] C.K. Chui, B. Kao, E. Lo, and R. Cheng. I/o-efficient algorithms for answering pattern-based aggregate queries in a sequence olap system. In CIKM, 2011.
- [CKRS04] S.S. Chawathe, V. Krishnamurthy, S. Ramachandran, and S. Sarma. Managing RFID data. pages 1189–1195, 2004.

References

- [CLKH09] C.K. Chui, E. Lo, B. Kao, andW.S. Ho. Supporting ranking pattern-based aggregate queries in sequence data cubes. In CIKM, 2009.
- [DGP+07] A.J. Demers, J. Gehrke, B. Panda, M. Riedewald, V. Sharma, and W.M. White. Cayuga: A general purpose event monitoring system. In CIDR, 2007.
- [GHL06a] H. Gonzalez, J. Han, and X. Li. FlowCube: constructing RFID flowcubes for multidimensional analysis of commodity flows. In VLDB, 2006.
- [GHL06b] H. Gonzalez, J. Han, and X. Li. Mining compressed commodity workflows from massive rfid data sets. CIKM, 2006.
- [GHLK06] H. Gonzalez, J. Han, X. Li, and D. Klabjan. Warehousing and analyzing massive RFID data sets. In ICDE, 2006.
- [HCD+05] J. Han, Y. Chen, G. Dong, J. Pei, B.W. Wah, J. Wang, and Y.D. Cai. Stream Cube: An architecture for multi-dimensional analysis of data streams. DPD, 18(2), 2005.
- [HWK+13] Z. He, P. Wong, B. Kao, E. Lo, and R. Cheng. Fast evaluation of iceberg patternbased aggregate queries. In CIKM, 2013.
- [HWK+17] Z. He, P. Wong, B. Kao, E. Lo, R. Cheng, and Z. Feng. Efficient pattern-based aggregation on sequence data. TKDE, 29(2), 2017.
- [KMWE14] C. Koncilia, T. Morzy, R. Wrembel, and J. Eder. Interval OLAP: analyzing interval data. In DaWaK, 2014.
- [KPW15] C. Koncilia, H. Pichler, and R. Wrembel. A generic data warehouse architecture for analyzing workflow logs. In ADBIS, 2015.
- [LKH+08] E. Lo, B. Kao,W.S. Ho, S.D. Lee, C.K. Chui, and D.W. Cheung. OLAP on sequence data. In SIGMOD, 2008.
- [LR10] M. Liu and E.A. Rundensteiner. Event sequence processing: new models and optimization techniques. In IDAR@SIGMOD, 2010.

References

- [LRG+10] M. Liu, E.A. Rundensteiner, K. Greenfield, C. Gupta, S.Wang, I. Ari, and A. Mehta. E-cube: Multi-dimensional event sequence processing using concept and pattern hierarchies. In ICDE, 2010.
- [LRG+11] M. Liu, E.A. Rundensteiner, K. Greenfield, C. Gupta, S.Wang, I. Ari, and A. Mehta. E-Cube: multi-dimensional event sequence analysis using hierarchical pattern query sharing. In SIGMOD, 2011.
- [LRR+10] M. Liu, M. Ray, E.A. Rundensteiner, D.J. Dougherty, C. Gupta, S.Wang, I. Ari, and A. Mehta. Processing nested complex sequence pattern queries over event streams. In DMSN, 2010.
- [LS03] A. Lerner and D. Shasha. AQuery: Query Language for Ordered Data, Optimization Techniques, and Experiments. In VLDB, 2003.
- [MKM+15a] P. Meisen, D. Keng, T. Meisen, M. Recchioni, and S. Jeschke. Bitmap-based online analytical processing of time interval data. In ITNG, 2015.
- [MKM+15b] P. Meisen, D. Kenig, T. Meisen, M. Recchioni, and S. Jeschke. TidaQL: a query language enabling on-line analytical processing of time interval data. In ICEIS, 2015.
- [MM00] H. Mannila and C. Meek. Global partial orders from sequential data. In KDD, 2000.
- [RDR+98] R. Ramakrishnan, D. Donjerkovic, A. Ranganathan, K.S. Beyer, and M. Krishnaprasad. SRQL: Sorted relational query language. In SSDBM, 1998.
- [RL10] C.W.W. Raymond and E. Lo. Competitive privacy: Secure analysis on integrated sequence data. In DASFAA, 2010.
- [SLR94] P. Seshadri, M. Livny, and R. Ramakrishnan. Sequence query processing. SIGMOD Rec., 23(2), 1994.
- [SLR95] P. Seshadri, M. Livny, and R. Ramakrishnan. SEQ: A model for sequence databases. In ICDE, 1995.
- [SLR96] P. Seshadri, M. Livny, and R. Ramakrishnan. The design and implementation of a sequence database system. In VLDB, 1996.

References

- [SZZA01a] R. Sadri, C. Zaniolo, A. Zarkesh, and J. Adibi. Optimization of sequence queries in database systems. In PODS, 2001.
- [SZZA01b] R. Sadri, C. Zaniolo, A.M. Zarkesh, and J. Adibi. A sequential pattern query language for supporting instant data mining for e-services. 2001.
- [SZZA04] R. Sadri, C. Zaniolo, A. Zarkesh, and J. Adibi. Expressing and optimizing sequence queries in database systems. TODS, 29(2), 2004.
- [vdA13] W.M.P. van der Aalst. Process cubes: Slicing, dicing, rolling up and drilling down event data for process mining. In AP-BPM, 2013.
- [ZJPL09] B. Zhou, D. Jiang, J. Pei, and H. Li. OLAP on search logs: an infrastructure supporting data-driven applications in search engines. In KDD, 2009.
- [ZKM13] Y. Zhang, M. Kersten, and S. Manegold. SciQL: Array data processing inside an RDBMS. In SIGMOD, 2013.