An Overview of Continuous Querying in (Modern) Data System

Riccardo Tommasini, INSA Lyon, CNRS Liris (France) Angela Bonifati, Lyon 1 University, CNRS Liris, IUF (France)

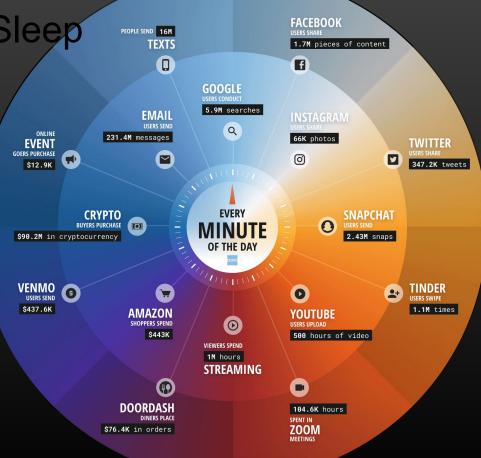




Slides at



Data Never Sleep





credits to Sara Robinson, Felipe Hoffa

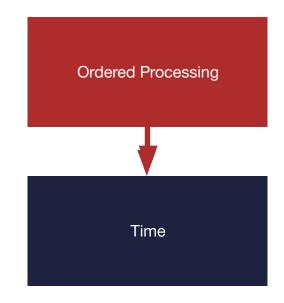
The Burden of Unboundedness

...requires a paradigm shift

Unboundedness

The Burden of

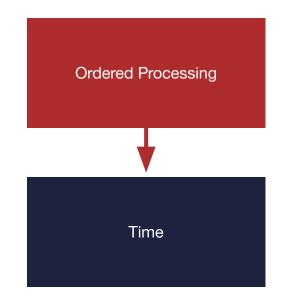
- A program expressed over an **infinite** input may **not terminate**
- Unless we can **reformulate** the notion of "**termination**"
- From an **infinite input** we may observe an **infinite output**
- We still need a way to **determine** what **part** of the **input** maps to the **output**



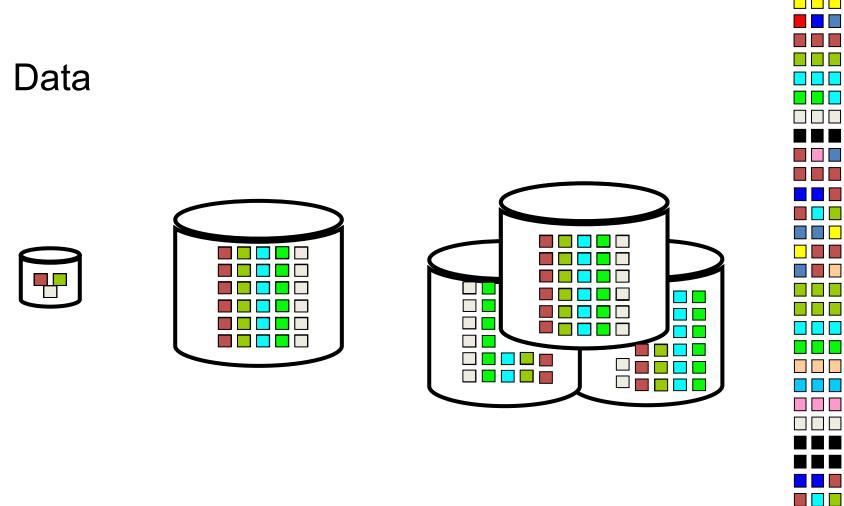
Unboundedness

The Burden of

- An **infinite dataset** poses the problem of where to start **computing**
- **Recency*** is a form of **temporality** that enables also **reactivity**
- Temporality** may assume other forms
 - About Time (Temporal Data)
 - Through Time (Versioned Data)
 - In-Time (Streaming Data)

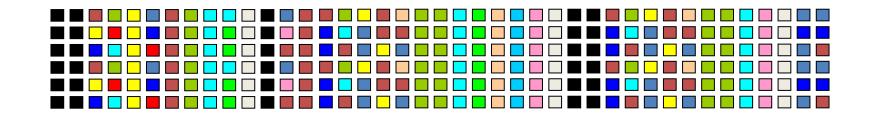


*Akidau, Tyler, et al. "The dataflow model: a practical approach to balancing correctness, latency, and cost in massive-scale, unbounded, out-of-order data processing." PVLDB (2015). ** Polleres, Axel, et al. "How does knowledge evolve in open knowledge graphs?." *Transactions on Graph Data and Knowledge* 1.1 (2023): 11-1.



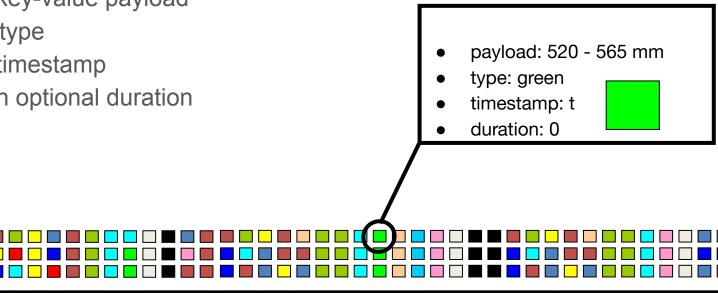
What is a Stream?

An unbounded partially ordered sequence of data points

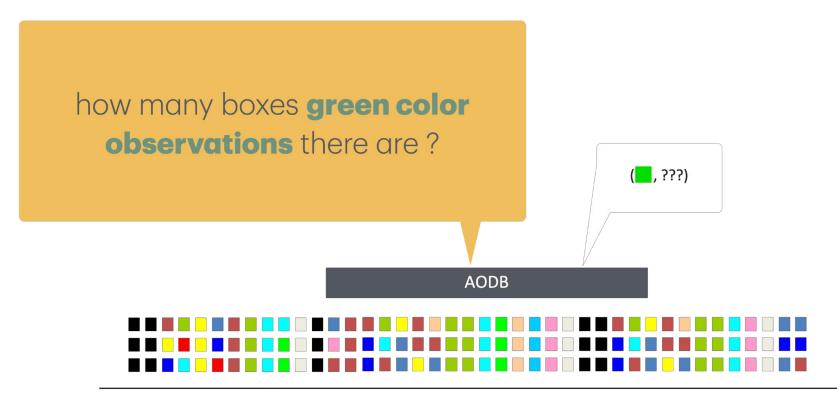


What is an Event?

- Event: time-based notification of a known fact defined by
- p a key-value payload
- τ , a type
- t, a timestamp
- d, an optional duration

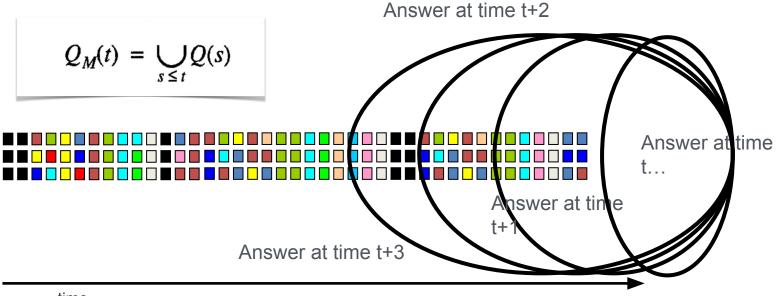


Continuous Queries on append only databases



Continuous Queries

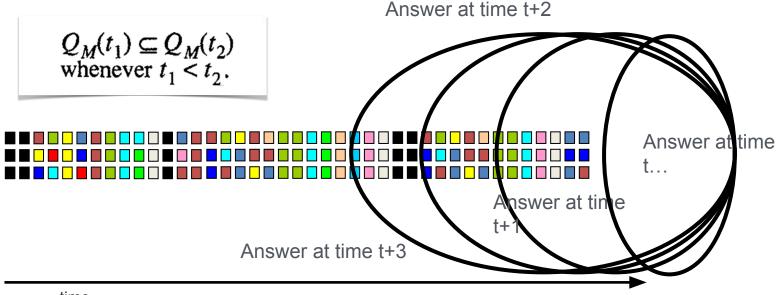
on append only databases



time

Continuous Queries (Monotonic)

on append only databases



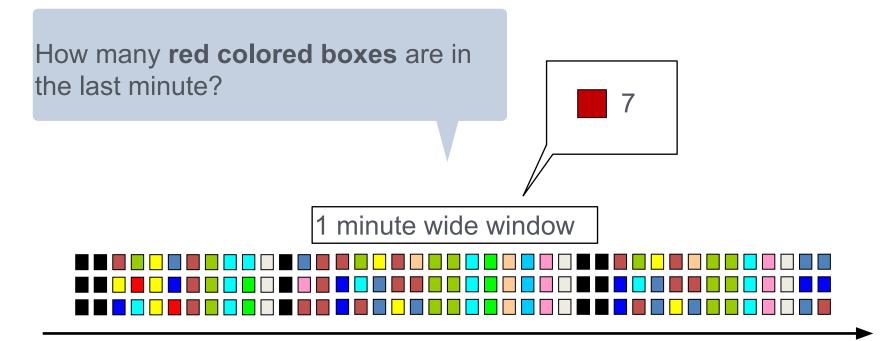
time



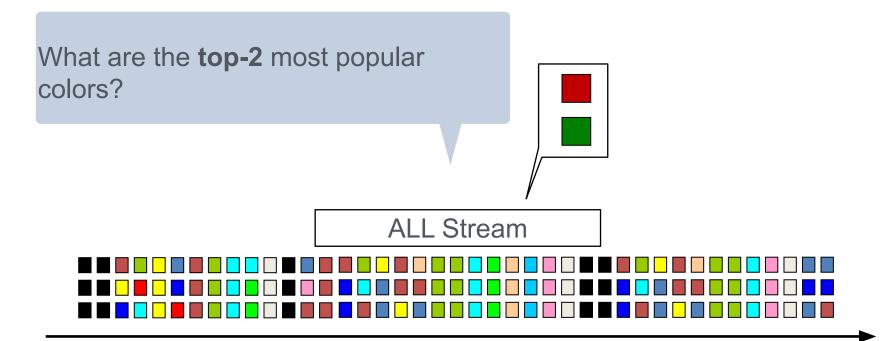
Monotonicity is not enough!

Can we achieve more?

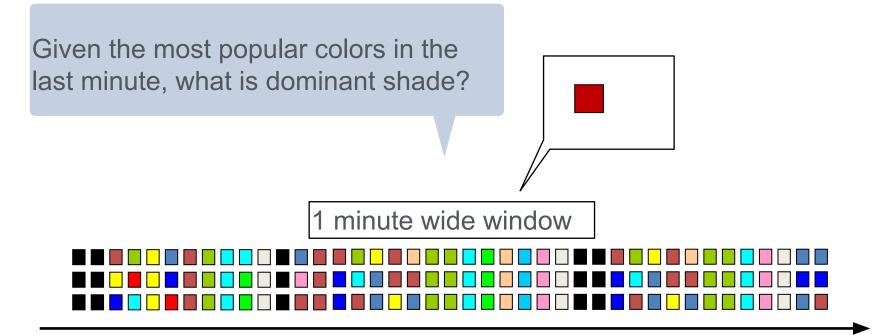
Continuous Aggregation



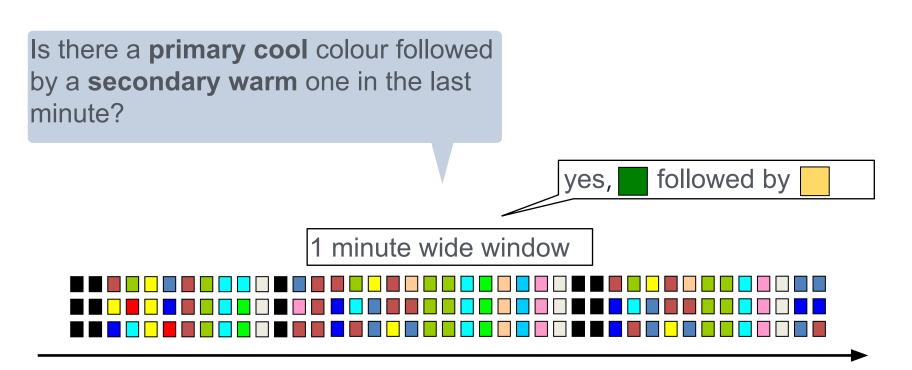




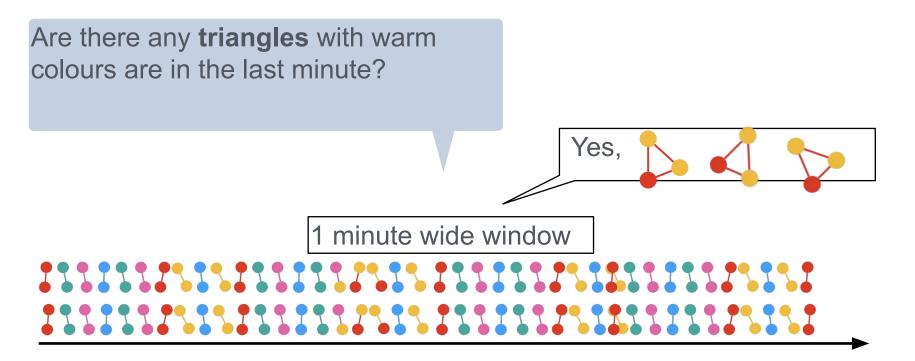
Skyline Continuous Queries



Complex Event Recognition



Graph Stream Processing



Historical Notes

Kramer, Yan-Nei,



Data Stream Management Systems

If you live near a stream

- Keep streams shaded. Trees and bushes keep the water cool for fish and help stabilize the banks. Do not remove streamside vegetation within 15 m of the stream.
- Keep litter and trash out of streams. Besides being unsightly, trash will collect into debris jams and block water flow. Limit in-stream cleanup activity to the summer months.
- Keep garden waste out of streams. Branches, grass clippings and weeds rot and reduce the amount of oxygen in the water.
- Keep pets away from streams. Animal waste is polluting. Pets entering streams can erode streambanks and cause siltation; their activity also disturbs wildlife and salmon living in streams.

Landscape with care. Despite good intentions, changes you make in and around streams may destroy spawning beds and fish habitat, or block fish migration. Do no build ponds, dams or bridges without guidance and approval from Fisheries and Oceans Canada and the B.C. Ministry of Environment.

The Solution

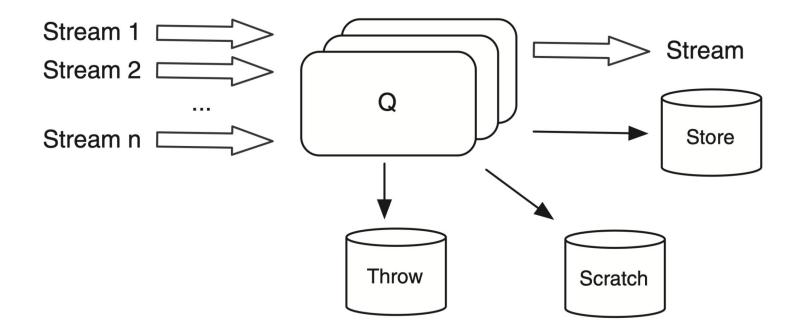
Everyone can take part in wise stream stewardship. Using these tips will help keep our streams healthy and beautiful now and for the future.

For more information, contact:

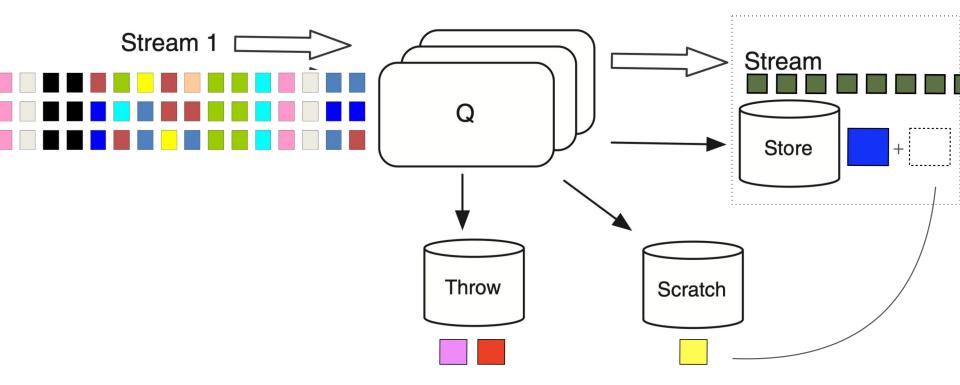
- Fisheries and Oceans Canada (604) 666-6614
- B.C. Ministry of Environment, Lands and Parks (604) 582-5200
- BC Recycling Hotline In the Lower Mainland: 732-9253 Toll-free: 1-800-667-4321

Fisheries and Oceans Pêches et Océans



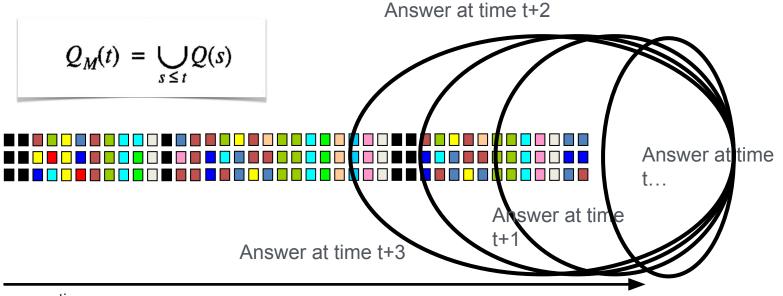






Continuous Queries

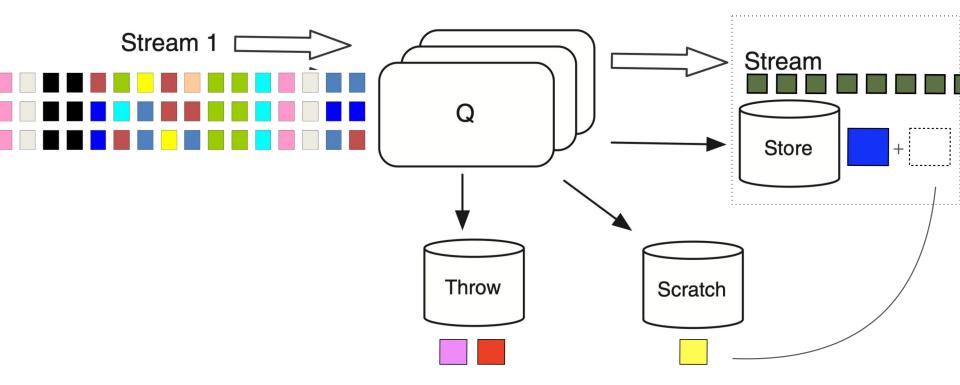
on append only databases



time

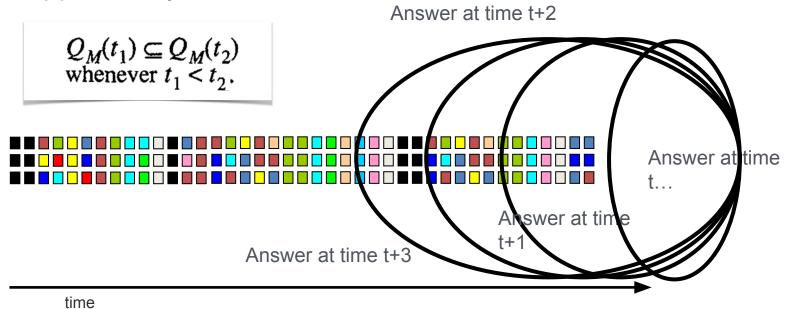
Terry, Douglas, et al. "Continuous queries over append-only databases." Acm Sigmod Record 21.2 (1992): 321-330.





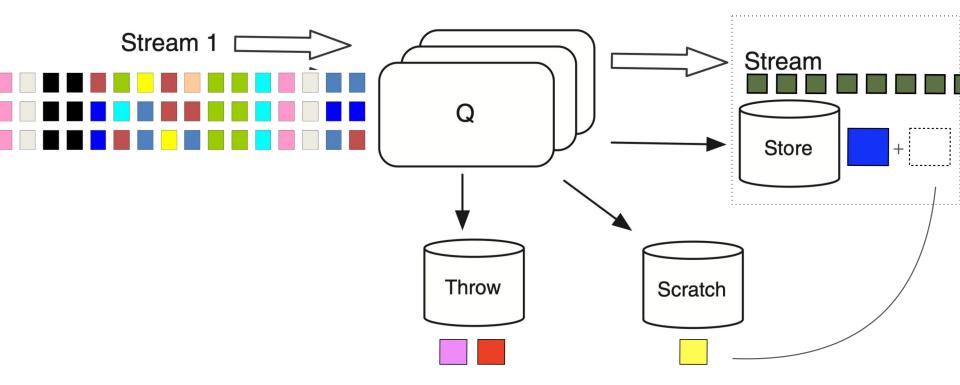
Continuous Queries (Monotonic)

on append only databases



Terry, Douglas, et al. "Continuous queries over append-only databases." Acm Sigmod Record 21.2 (1992): 321-330.





Monotonicity Explained

- **Monotonic** queries produce an **append-only** output stream and therefore do not incur deletions from their answer set.
 - A query is monotonic if for two instances of the database S1 and S2 such that $S1 \subseteq S2$ then $Q(S1) \subseteq Q(S2)$, where Q(Si) denotes the set of tuples that satisfy Q when applied to the instance Si.
- Only stateless **operators** over infinite streams (projection, selection, time-wise union, and distributive aggregates) can give rise to **monotonic** queries.
- Hence, non **monotonicity** is caused by so called **blocking** operators, i.e., operators that need to see the whole stream to report

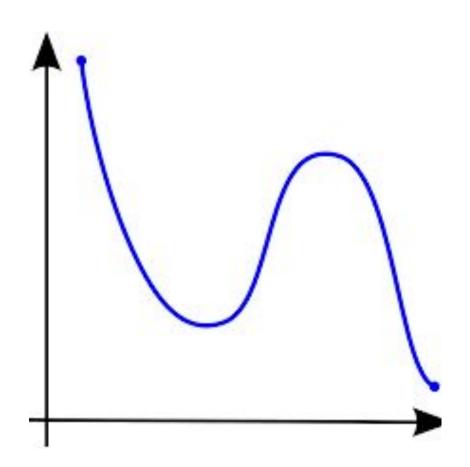
Non-monotonicity

Causes of

The concept has taken many names

- stateless/stateful functions
- blocking/non blocking operators
- event-level/stream-level semantics

All notions share the intuition of "memory", how far do I have to know to answer?



Interval Strategy

- If there are no deletions in the database, the answer to a continuous, non-monotonic query Q can be approximated as Q = P – N
- $P(k\tau) P((k-1)\tau)$ at every interval $[(k-1)\tau, k\tau)$
- In the worst case, this approximation gives a superset of set of data items in the right hand side of the equation below

$$Q(k au) - Q((k-1) au) = \bigcup_{t_0 \leq s \leq k au} Q(s) - \bigcup_{t_0 \leq s \leq (k-1) au} Q(s).$$

Barbará, Daniel. "The characterization of continuous queries." International Journal of Cooperative Information Systems 8.04 (1999): 295-323.

Fixed-Structure Rewriting

- A rewriting Q = P N is fixed structure if the following conditions are true:
- P is monotonic
- Interval strategy holds while
 - P holding within k1 and Inf, N holding within k2,k3, with k1 < k2

$$P(k\tau) - P((k-1)\tau) = \bigcup_{t_0 \le s \le k\tau} Q(s) - \bigcup_{t_0 \le s \le (k-1)\tau} Q(s).$$

Barbará, Daniel. "The characterization of continuous queries." International Journal of Cooperative Information Systems 8.04 (1999): 295-323.

Admitting Deletions (append-only dbs)

- A query Q is deletion sensitive iff for S1 and S2 such that S2 ⊆ Si, then there exists an item D1 ∈ S1, D1 ∈ S2 such that one of the following is true:
- D1 ∈ Q(S1)(t) AND D1 ∉ Q(S2)(t)
- $D1 \notin Q(Si)(t) AND D1 \in Q(S2)(t)$

The semantics evolves as

$$P(k\tau) - P((k-1)\tau) = \bigcup_{t_0 \le s \le k\tau} Q(s) - \bigcup_{t_0 \le s \le (k-1)\tau} Q(s) - D_{\tau}^k.$$

Barbará, Daniel. "The characterization of continuous queries." International Journal of Cooperative Information Systems (1999)

Fix the Size of the Answer

Fixing the Size of the Answer

Idea: summarize the characteristics of a stream reducing the memory footprint

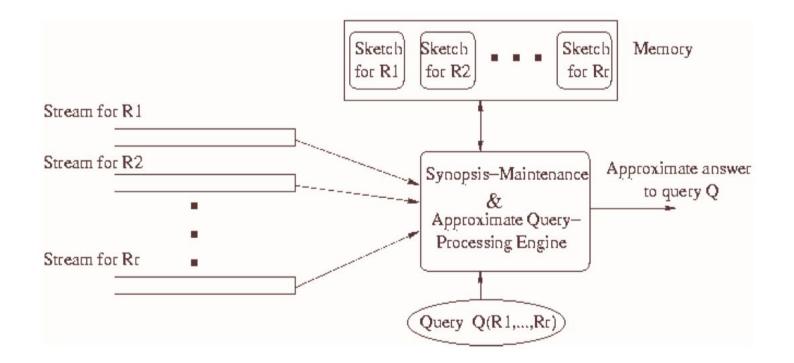
A **histograms** summarize a dataset by grouping the data values into buckets and compute for each bucket a set of summary statistics

Wavelet transform the data to represent the most significant features in a frequency domain

Sketches, data structures or algorithms that provide approximate answers to given queries.

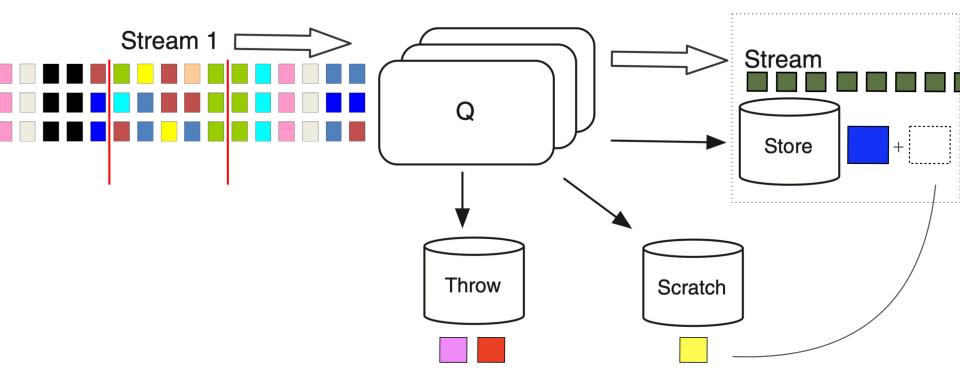
Babcock, Brian, et al. "Models and issues in data stream systems." *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. 2002.

Synopsis-Based DSMS



Modify the Input Stream

CQ+Pubctuations@DBMS



Punctuations

- A punctuation is a **predicate** on stream elements that **must** evaluate to **false** for **every element** following the punctuation (Boolean functions).
- In the **original** papers, they are presented as a simple **grammar**
 - 0 [*,+,value,[], range]
 - A tuple matches the punctuation if each of its attributes matches the corresponding pattern
- A punctuated stream is a data stream that contains additional information describing a (possibly empty) subset of data over the domain of the stream

Punctuations Correctness

- A **punctuated stream** S is **grammatical** if for all i, for all j > i, if the punctuation p ∈ S[i] □ and the tuple t ∈ S[i -> j], t does not match p.
- **Safety**: That is, we never emit output unless we can be sure it will not conflict with any later input.
- **Completeness**: we always emit an output if it will necessarily be generated by the relational operator under any additional input, including no input.

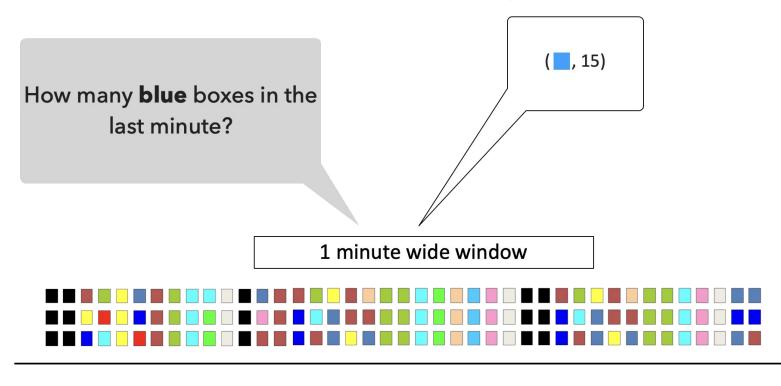
Advantages/Caveats of Punctuations

- Simple to implement
- All **result** data items for a query will eventually be output.
- **Cleanses**. Every data item that resides in the state for any operator in the query **will** eventually be removed.

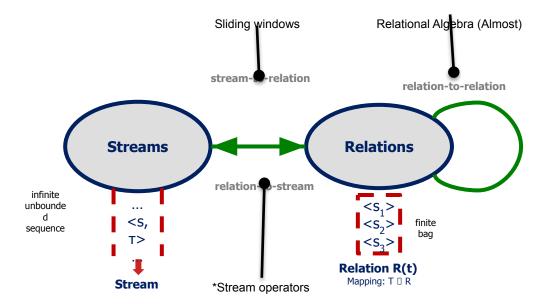
- Who is going to emit the punctuation?
 - Sources?
 - Other operators?
- Different data models have different punctuation semantics
- We still do not know what queries benefits from a given punctuation

Modify the Output Stream a.k.a. Modify the Query

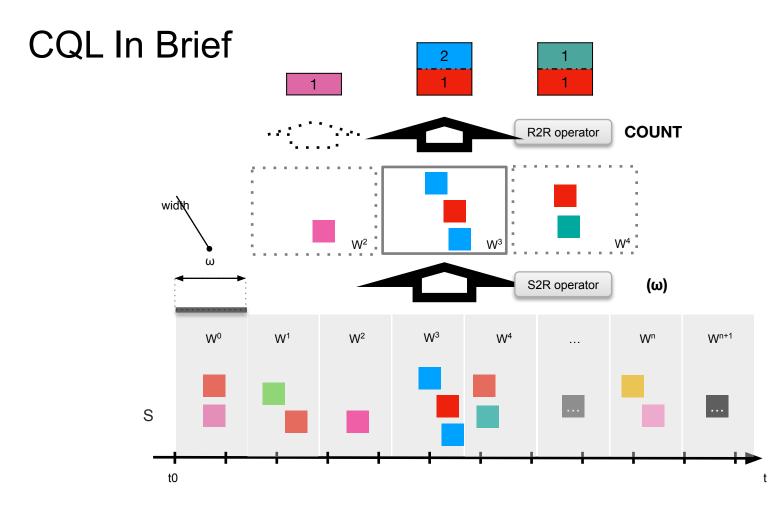
Window-Based Continuous Querying



Continuous Query Language

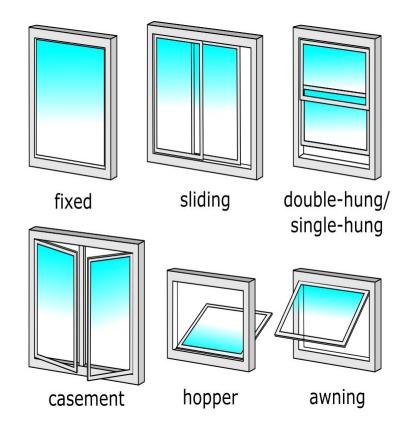


Arasu, Arvind, Shivnath Babu, and Jennifer Widom. "The CQL continuous query language: semantic foundations and query execution." The VLDB Journal 15 (2006):.



Types of Windows

Window Types	Parameters
Sliding	width
Hopping	Width, slide
Tumbling	Width == slides
Session	Inactivity



Verwiebe, Juliane, et al. "Survey of window types for aggregation in stream processing systems." The VLDB Journal 32.5 (2023): 985-1011.

Windows and Monotonicity

- Weakest non-monotonic queries do not store state and do not reorder incoming tuples during processing; tuples are either dropped or appended to the output stream immediately.
 - Projection and selection over a single sliding window are weakest non-monotonic
- Weak non-monotonic have the property that the expiration time of each result tuple can be determined without generating negative tuples on the output stream.
 - join, duplicate elimination, and groupby.
- **Strict non-monotonic** queries have the property that at least some of their results expire at unpredictable times.
 - Negation over two windows is one example.

Golab, Lukasz, and M. Tamer Özsu. "Update-pattern-aware modeling and processing of continuous queries." *Proceedings of the 2005* ACM SIGMOD international conference on Management of data. 2005.

Advantages/Caveats of Windows

- SP over sliding Windows are very easy to optimise
- Advantages for parallelised computations
- Enable efficient aggregation and possibly synopsis [paper]

- Users need to know the data semantics
 - Which may be hard given their non-monotonic behaviour
- Out-of-order processing requires sophisticated strategies
- Currently are system-dependent, and harm query portability

Golab, Lukasz, and M. Tamer Özsu. "Update-pattern-aware modeling and processing of continuous queries." *Proceedings of the 2005* 47 ACM SIGMOD international conference on Management of data. 2005.

Correctness

In Search of

IN SEARCH OF LOST TIME

MARCEL PROUST



A Temporal Foundation

Data Model

Physical Stream (PS) is infinite sequence of tuples (e, [ts, te)) with the same schema. Two elements i, j, are value-equivalent iff ei==ej

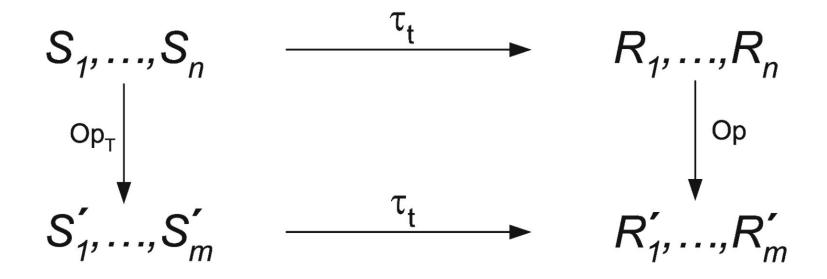
Logical Stream (LS) is a possibly infinite multiset of triples (e, t, n) composed of a record $e \in \Omega$, a point in time $t \in T$, and a multiplicity $n \in N$.

Physical to Logical (**planning**): For each tuple (e, [ts, te)) \in LS, we split the associated time interval into points of time at finest time granularity.

Logical to Physical (execution)

- Map each logical stream element (e,t,n) into a physical element (e,[t,t+1))
- Coalesce value-equivalent elements that are close to each other (maximal validity)

Snapshot Reducibility



Krämer, Jürgen, and Bernhard Seeger. A temporal foundation for continuous queries over data streams. Univ., 2004.

Snapshot Reducibility

For a given logical stream LS and a specified point in time t, the timeslice operation returns a non-temporal multiset of all records in LS that are valid at time instant t

(Snapshot-Reducibility) A **logical** stream operator **opT** is snapshot-**reducible** to its non-temporal counterpart **op** over multisets, if for any point in time $t \in T$ and for all logical input streams LS_1, ..., LS_n $n \in SI$,

Operators over logical streams

Kramer et al introduce the following operations on logical stream: filter(σ), map (μ), Cartesian product (×), duplicate elimination (δ), difference (–), group (γ), aggregation (α), union (\cup) and window (ω).

Unfortunately, windows, unions, and are again non-window reducible...

Are we condemned to observe this kinds of results...?

The course of querying streams

Several people tries to overcome the issue of "memory"...

Yan-Nei et al showed us that if we use **sequences as our basic data model**, non-monotonic queries become so dominant that only basic project/select operations can be expressed as continuous queries.

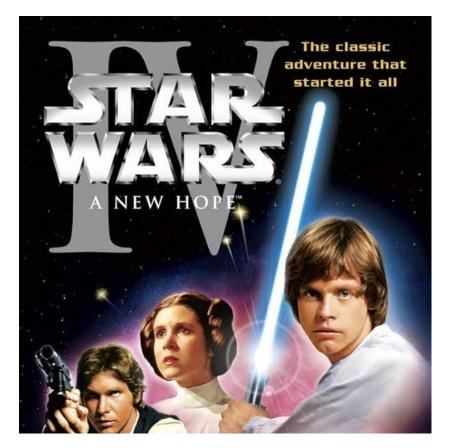
The non-blocking subset of relational algebra (NB-RA) and SQL (NB-SQL) are not NB-complete i.e., it cannot express every monotonic set function

Law, Yan-Nei, Haixun Wang, and Carlo Zaniolo. "Relational languages and data models for continuous queries on sequences and data 54 streams." *ACM Transactions on Database Systems (TODS)* 36.2 (2011): 1-32.

A New Hope User defined Aggregates

A query language that supports non-blocking UDAs and set union can express all monotonic set functions on data streams.

While **UDAs** makes the query language NB-complete they also make it **turing complete** on classical tables



Law, Yan-Nei, Haixun Wang, and Carlo Zaniolo. "Relational languages and data models for continuous queries on sequences and data 55 streams." *ACM Transactions on Database Systems (TODS)* 36.2 (2011): 1-32.

Time and Approximation

However, when moving to timestamped data stream (ordered set), we are losing again the monotonicity of binary operators (.e.g, join).

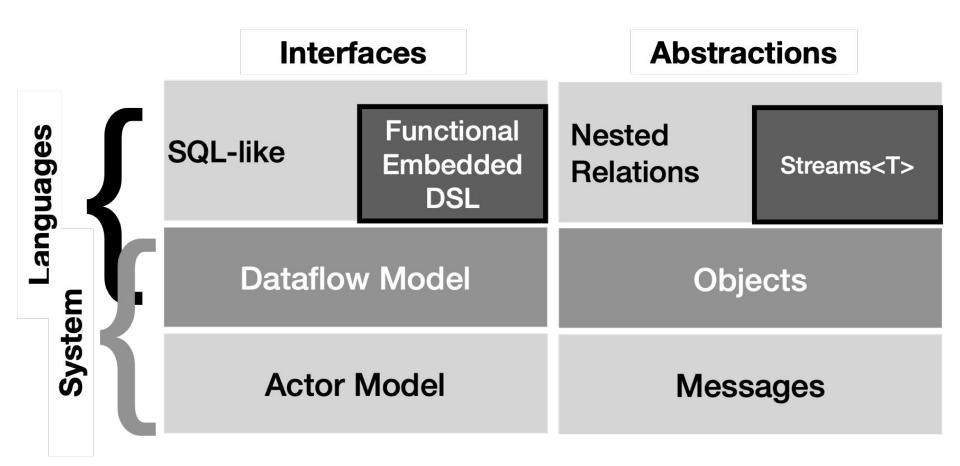
- Can at least have a monotonic approximation?
 - Yes, if we consider the recent sub-portion of the data stream: t-operations (union, product, difference) only look at the data up to t
- Focusing on streams that have no delay (not true in practice):
 - a query language that has UDA and a version of union that is **t-approximated** is NB-complete.

Law, Yan-Nei, Haixun Wang, and Carlo Zaniolo. "Relational languages and data models for continuous queries on sequences and data streams." *ACM Transactions on Database Systems (TODS)* 36.2 (2011): 1-32.

Streaming Systems

For Big Data





Continuous Queries



SQL-Like Languages

- New trend is hiding the complexity of the processing behind SQL
- Alternative design debates on how to extend the languages
 - \bigcirc WINDOW Clauses
 - **CEP** Operations
 - Extended GroupBy
 - Ο Report Controlling

One SOL to Rule Them All: An Efficient and Syntactically Idiomatic Approach to **Management of Streams and Tables**

An Industrial Paper Tyler Akidau

Seattle, WA, USA

Kathrvn Knight

Edmon Begoli Oak Ridge National Laboratory / Apache Calcite Oak Ridge, Tennessee, USA begoli@apache.org

Iulian Hvde Looker Inc. / Apache Calcite

jhyde@apache.org

Oak Ridge National Laboratory San Francisco, California, USA Oak Ridge, Tennessee, USA knightke@ornl.gov

ABSTRACT

Real-time data analysis and management are increasingly critical for today's businesses. SQL is the de facto lingua franca for these endeavors, yet support for robust streaming analysis and management with SQL remains limited. Many approaches restrict semantics to a reduced subset of features and/or require a suite of non-standard constructs. Additionally, use of event timestamps to provide native support for analyzing events according to when they actually occurred is not pervasive, and often comes with important limitations.

We present a three-part proposal for integrating robust streaming into the SQL standard, namely: (1) time-varying relations as a foundation for classical tables as well as streaming data. (2) event time semantics. (3) a limited set of optional keyword extensions to control the materialization of timevarving query results. Motivated and illustrated using examples and lessons learned from implementations in Apache Calcite, Apache Flink, and Apache Beam, we show how with these minimal additions it is possible to utilize the complete suite of standard SQL semantics to perform robust stream processing.

Publication rights licensed to ACM. ACM acknowledges that this contribu-

Fabian Hueske Google Inc. / Apache Beam Ververica / Apache Flink Berlin, Germany takidau@apache.org fhueske@apache.org

Kenneth Knowles

Google Inc. / Apache Beam Seattle, WA, USA kenn@apache.org

CCS CONCEPTS

 Information systems → Stream management; Query languages;

KEYWORDS

stream processing, data management, query processing

ACM Reference Format:

Edmon Begoli, Tyler Akidau, Fabian Hueske, Julian Hyde, Kathryn Knight, and Kenneth Knowles. 2019. One SQL to Rule Them All: An Efficient and Syntactically Idiomatic Approach to Management of Streams and Tables: An Industrial Paper. In 2019 International Conference on Management of Data (SIGMOD '19). June 30-July 5. 2019, Amsterdam, Netherlands, ACM, New York, NY, USA, 16 pages, https://doi.org/10.1145/3299869.3314040

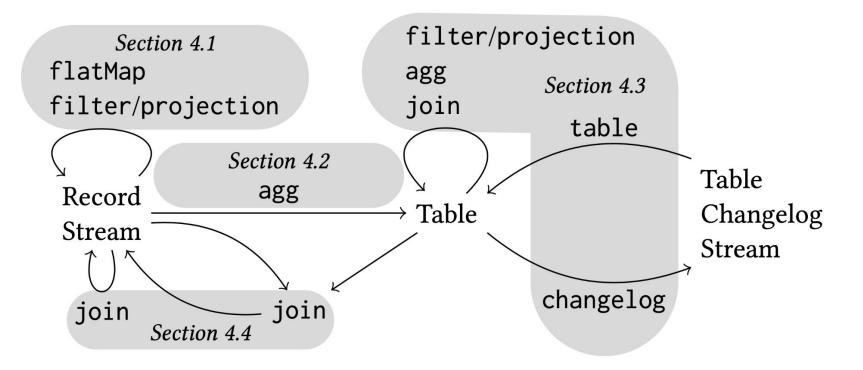
1 INTRODUCTION

The thesis of this paper, supported by experience developing large open-source frameworks supporting real-world streaming use cases, is that the SQL language and relational model, as-is and with minor non-intrusive extensions, can be very effective for manipulation of streaming data.

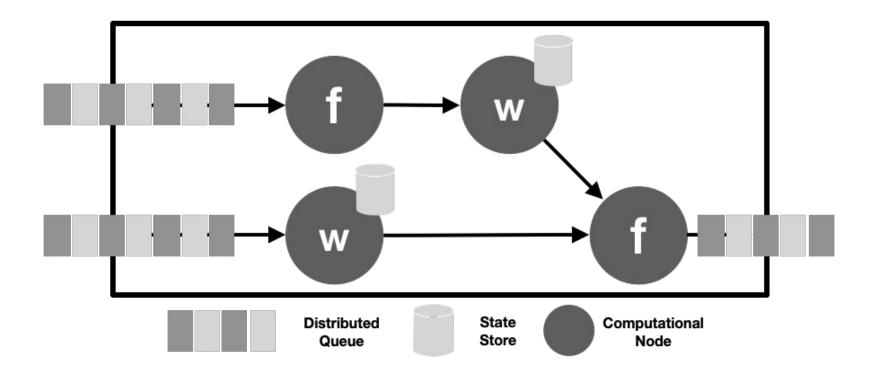
Our motivation is two-fold First we want to share our ob-

60

Functional DSL



Operator Topology



Continuous Queries

in the

Modern Data Landscape

THE 2023 MAD (MACHINE LEARNING, ARTIFICIAL INTELLIGENCE & DATA) LANDSCAPE

INFRASTRUCTURE	ANALYTICS	MACHINE LEARNING & ARTIFICIAL INTELLIGENCE	APPLICATIONS – ENTERPRISE
STORAGE MPP DBS DATA AVESS / DA	BIPATFORMS SLOOKER SLOOKER SLOOKER SUSSEE SU	Data science Data science Pata science<	
		Contention Content	Portuge P
STACE STACES MARGINE Second Strategy States States Second Strategy States States Second Strategy States States Second Strategy States States States Second Strategy States	Construction of the second sec	Sitemento indexento indexe	APPLICATIONS - HORIZONTAL
DATA TRANSFORMATION → Census → Census → Consus → Co			Construction C
Arestrate to the state of	history excessors Martory excessors Martor thr Ouery ENCIRE ENCIRE Name AppOyromics AppO	A HARDWARE	GE FAINCE A HEALTHCARE UFSCREETS TRANSPORTATION ADDRUIDE NOUTSELL & OUT A TANK AND A TA
A 373 200400 Present dated Construction Street Construction Street C	Grand Grand Grand Grand Image: Strain and St	Device Grant Control Cont	Market File Service

OPEN SOURCE INFRASTRUCTURE					
wyn Caubernates Charl an	MISSIGNIG MILLING Strain Strain Control Contr	Al MODELS & ARCHITECURES Gampy Darking Scarg Carry Lange Gampy Darking Scarg Carry Lange Concert International Carry Carry Carry Concert Carry Concert Market Diversion International Carry Concert Carry Concert Market Diversion International Carry Concert Carry Car			

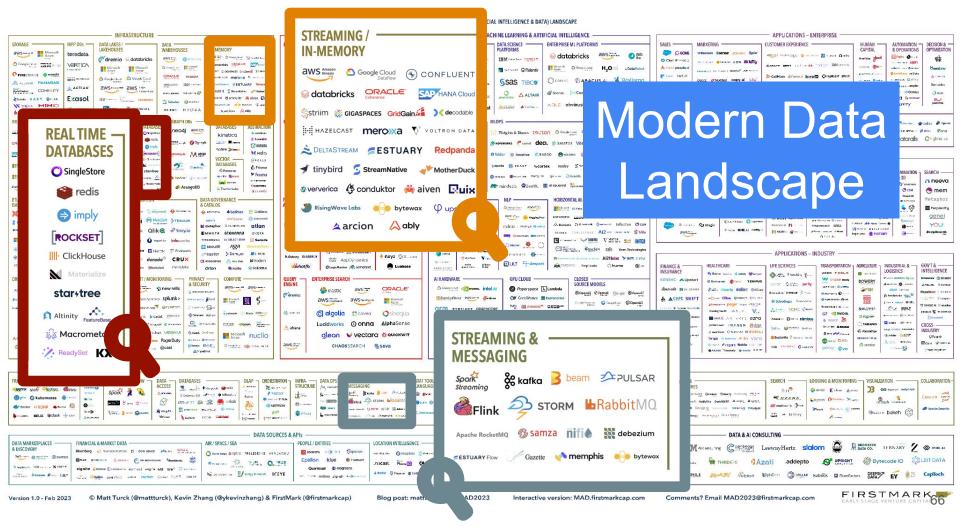
DATA SOURCES	& APIs	DATA & AI CONSULTING
DATA MARKETPACES 8 DISCOVERY S DISCOVERY	Cepsilon kiue • Hartenst Jnacast. Piece • • • • • • • • • • • • • • • • • •	

Version 1.0 - Feb 2023 © Matt Turck (@mattturck), Kevin Zhang (@ykevinzhang) & FirstMark (@firstmarkcap) Blog post: mattturck.com/MAD2023

Interactive version: MAD.firstmarkcap.com

THE 2023 MAD (MACHINE LEARNING, ARTIFICIAL INTELLIGENCE & DATA) LANDSCAPE

INFRASTRUCTURE	ANALYTICS	MACHINE LEARNING & ARTIFICIAL INTELLIGENCE	APPLICATIONS – ENTERPRISE
STORAGE MPP DBs DATA LAKES / DATA	BI PLATFORMS VISUALIZATION	DATA SCIENCE DATA SCIENCE ENTERPRISE ML PLATFORMS	SALES MARKETING CUSTOMER EXPERIENCE HUMAN AUTOMATION DECISION & CAPITAL & OPERATIONS OPTIMIZATION
awsing and analysis and analysi	SLooker 🐡 👐 🏥 + a ble a u 📑 Microsoft	Suppler == 00 0 IBM Outler @ databricks aws the outle	To CAR Statement of Service and the Carter of the Service and
	Barrier BM Service Google		A CARLEN AND A CAR
C BEING STORAGE (B WERKER) C BEING STORAGE SCHOOLS	Astrato ili plotiy Press		Prest Tablad
ALLIAND PANASAS	Soda Gaisense Arscalle	Sas TIRCO COMING HIABACUS.AI & Wallaroo	Among Cinded Streads Control Description
COHESITY 32 ACTIVAT CONSTANT OF STATES OF STAT		ALTAIR @ snares >Continues @dotData Modular	stories Gauge and Southant Sou
Quarter VAST OLLAR EXCEOI 2017-2017 CLOREDER CLOREDER CLOREDER STREAMER CLOREDER	CROW 6 ky/cts (c) heave		Storme Options Phrosee Willing and Phrosee Willin
WIEKA MINIO Stankling Accounting Contraction Contraction (Contraction Contraction Contrac	POPSQL #Lightdash 🛛 🕊 Streem it. 🌵 Rill ≪ cumul.lo	Curvenote >Nextjournal	LEGAL PARTNERSHIPS REGTECH & FINANCE
RDBMS - NoSOL DATABASES - NewSOL DATABASES - REAL TIME - GRAPH DBs - GPU - DATABASE -	DATA ANALYST PLATFORMS CUSTOMER DATA	DATA GENERATION MLOPS	COMPLIANCE
DATABASES DATABASES ABSTRACTION	PLATFORMS	& LABELING	Starten ber of Besidenty Areas Star Starting Court Starting
An art of the second se	STEALIUM S- ACTIONS		reveal of the statement determines where the statement of the statement determines pilot takement determines On phases
CRACLE CARE AND CONCERNING CONCER	> pyramid 🚺 🏪 2420000 🗇 Segment 🗮 Simon	Upwork apparted to the average of the second deci. B base ten Verta (ruera @ toutre trusted)	Hand Carter address of State
STARC MCTRAIL AMAINT Freedow Paramer AMAIN	guavus 🏟 Artable @ Determen 🗊 = = : : : : : : : : : : : : : : : : :	oppen Ounity Olidder & iterative CRASCO Owners Cales Cares Arrize 75	APPLICATIONS – HORIZONTAL
	anod ⁰ t 💦 saak 🏦 i za cu 🏶 cube Stylks 🗣 🔊 🔍 Orderande	Sand Annual Sand Sand Sand Sand Sand	APPLICATIONS - HORIZONIAL
Contract of the second	Peon - Equals glan Bing Ger Alastin optimova		CODE & TEXT AUDIO & VOICE IMAGE VIDEO EDITING ANIMATION SEARCH
	👰 🖹 Lide W Werkstreamle 🖉 carryos 🚺 🏙 🦛 d ruddandwk	KUI V7 PolategenLESON	DOCUMENTATION Security Defension Security Defen
answer Speedo Speedo ArangoDE Trabelalas approver	Deephoven VWeld Trevius Standard DUBCONIC CosyNCARI	vastive 7985 grelet Mage and mindsob @ZeelML @ alsquared \$20000 @DUST #finegrain	Security Childs Constant Statements Constant London Oregonal Viola Oregonal
ETL/ELT/ DATA INTEGRATION DATA GOVERNANCE	PRODUCT LOG ANALYTICS CRYPTD / WEB 3	COMPUTER SPEECH NLP HORIZONTAL AI / AGI	Constant Con
DATA TRANSFORMATION	ANALYTICS splunks 🧟 Oracitation ANALYTICS	VISION Sin 000 Despetar destriction Official Official Official Constant Constant Official	
X dbt talend atteryx	Sumo logic salarving	a teagra and a subject and a s	Recent Out a second while special with special second on toxes the second genei
	Tissanci Ppendo 🛛 kbana Silogalio ELLIPTIE	actives and a second and a second a sec	Bracer O Magin Contract Package of Package o
ementer: kleene (1) and a materialk emetodik Qlik@ d'traylo	Contentsquare MEZMD Coralogix XTRM	The second secon	awstraam + wax one at memory in the analysis of the second stand because the second stand sta
Koveri @ Aktyse (C) Megnetics Attacked and Aktyse (C) Megnetics Attacked and Aktyse (C) Megnetics (C) Megneti	I Heap Some Water Deserve Avantin Anansen		
tomr beliketake O ∞ # R Rivery Areivitakie Biskewelow E Hattle Fredepoist @ castor Demust Biskeke	Educe Clarger	Converse States A American American A Americ	APPLICATIONS - INDUSTRY
Security Courses Courses Security denodes CRUX Categories Annual MVcdem	ease AppDynamics	Alter Var. Active A	FINANCE & HEALTHCARE LIFE SCIENCES TRANSPORTATION _ AGRICUTURE _ INDUSTRIAL & GOV'T &
Matter and rentery area and a solutions and a solution and a solu	ulluhotfix OlugieHurise 🖓 scriptext 📄 Lubbees	These states we wanted the second sec	INSURANCE PRACTICAL CONTRACT OF STORE CONTRACT CONTRACT OF A CONTRACT OF
ORCHESTRATION DATA QUALITY & FULLY MGMT / MONITORING PRIVACY COMPUTE	QUERY ENTERPRISE SEARCH	AI HARDWARE GPU CLOUD CLOSED SOURCE MODELS	KENSHO A Upstart SERVICE TO A LOSS OF THE DASK
ASTRIMUMER OBSERVABILITY MANAGED AWS/ Onew relic ASTRIMUMER OF A CONTRACT OF A CONTRAC		Cooperand Cooperand Cooperand	Service Contract Cont
Present diarest direction and another W42 100 Applyments splunks and and and and an	Caremia aWS (tational aWS (support and Azuro	Stantigational Statements Construction	A CAPE SHIFT
Diamana union accelerate () metaplane () Phillippine III starwinds ally vitters Charteries Tere	Lyretaa	OLUM SYNTIANT GRAPHCORE FUS MI MORGIC" GRIDAN	Transfer States tourse tourset
BWStertam - affet Rieman Av. Staboola O tabrik > tribt Vitvoiri Ohmene Syraa Strand Vitvoiri	🕸 🖛 🔤 🕼 🖓 🖓 🖓 🖓 🖓 Sinequa		
Marcett Lalend SOBA: () Rove // Nexto Provent () Surger (Annual SoBa (A ahana Lucidworks O onno AlphaSense		ALTER ARACEA CTARANES STARS OF CROSS
Street Street Wesself VESAV Street Street Provide August VESAV	oleon Vectara & QUICKWIT		
And the Alexandro and Alexandr	aws CHAOSSEARCH @seva	STREAMING &	
Construction Const	Chrossevich (Useva	MECCACINIC	Access Tansaty & saids Tansaty
		L MESSAGING	
FRAMEWORKS FORMAT QUERY/DATA FLOW DATA DATABASES OLAP ORDES	STRATION - INFRA DATA OPS	spark & kafka 3 beam 2 PULSAF	R SEARCH LOGGING & MONITORING VISUALIZATION COLLABORATION
ACCESS S Annu Smith	STRUCTURE A variety MESSAGING ANG	Succining 0	and a static E show A ways 3 00 server match tob
Der Diebergerten Barren Sport & Sport & Sport P			string Grand Charge training theirs
🖝 declar - V CROP 🚓 👘 🚳 🚳 📾 🔤 🖉 Same de la 🖉 🖉 25 alte 🔉 🚔 Milak 🚲 👘	er Y nue Carlos	🚆 🖓 Flink 🧦 sтокм 📙 Rabbit М(refination beinger Avery Areth. // mature h fre
Annual Annua	Series L mailman	📰 🍘 🖓 STORM 🗳 Rabbit MC	A Googe A secure strange stran
Woher beite unter joiners joiners joiners joiners joiners joiners	the Kan		2
D474 COL	JRCES & APIs	Apache RocketMQ 🔞 Samza 🏢 🎆 debeziun	n DATA & AI CONSULTING
DATA MARKETPLACES FINANCIAL & MARKET DATA AIR / SPACE / SEA	PEOPLE / ENTITIES LOCATION INTELLIGENCE	Apacite Rocketing	
			🖬 al Consulting Computing LeewayHertz sicilom 🚇 🖟 Barra co. I FRNARY 🙎 🕸 Inite al
TATION & DAWEX		Gazette 💊 memphis 🔶 bytew	ax 🙀 THIRDEYE Azoti .addepto 🔗 UPRIGHT 🖉 Bytecode IO 💲 LEIT DATA
Ywekeen Illing and North Kinghitz Grant Ecological earriest wight MX tink? Harden	Quanteast Cognism		
and the second s	CEYE or total melissa zignali deditosituti A Ender @Piese.ai @cust		MILE 2008.87 Subjust UphDate Dellar kubrick I fuenfactor Data EY #25 CapTech
 Biotoxi, Laborati, Statulari, Trata, Tata, Tata, 14 			
Version 1.0 - Feb 2023 © Matt Turck (@mattturck), Kevin Zhang (@ykevinzhang) & I	FirstMark (@firstmarkcap) Blog post: matt	AD2023 Interactive version: MAD.firstmarkcap.com	Comments? Email MAD2023@firstmarkcap.com
			EARLY STAGE VENTURE CAPITAL
		-	65



Modern Landscape

- Real-Time Databases
 - focus on OLAP
 - long time series
- Streaming Databases
 - in-database windowing
 - continuous computation



time**plus**

MATERIALIZE









Incremental View Maintenance a long standing problem

- Beyond conjunctive queries, there are studies on IVM for intersection joins, Datalog, Differential Datalog, and DBSP (best paper VLDB 2023)
- The maintenance of complex analytics over evolving databases, which includes linear algebra computation, collection programming, and in-database machine learning.





MATERIALIZE

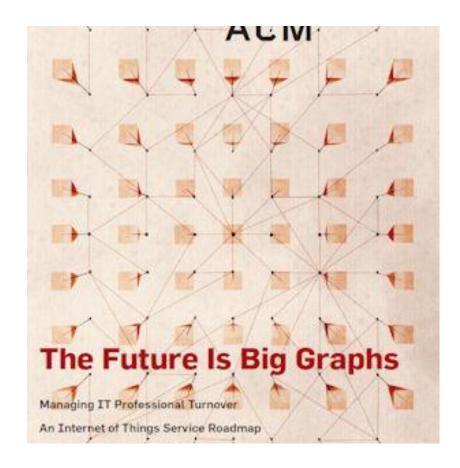
Feldera

Streaming Graphs

The Future is Big Graphs

In 2021 already

- A Community Vision of the role of graph in the future years
- Streaming Graph processing is core
 - Complex Query Execution
 - Incremental Graph Analytics
 - Temporal Dynamic Graphs



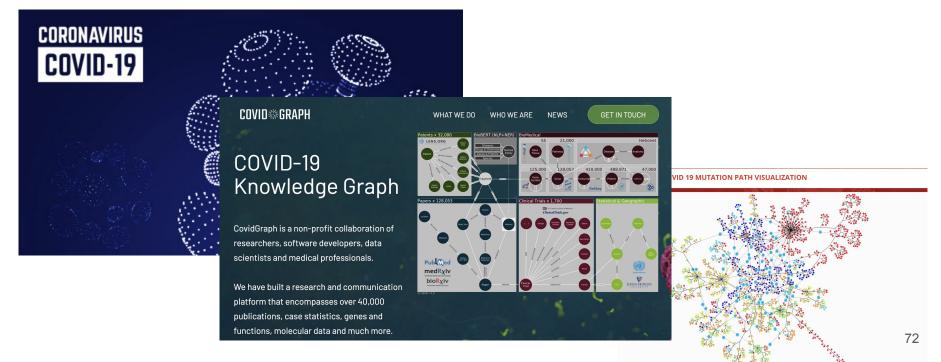
Challenges for Next-generation Graph Processing Systems

- Ch1. A lattice of graph data models and graph algebras
- Ch2. Complex data management ecosystems
- Ch3. Performance and benchmarking

S. Sakr, A. Bonifati, H. Voigt, A. Iosup et al. "The Future is Big Graphs: A Community View on Graph Processing Systems" Commun. ACM 64(9): 62-71 (2021)

Graphs are ubiquitous across diverse applications

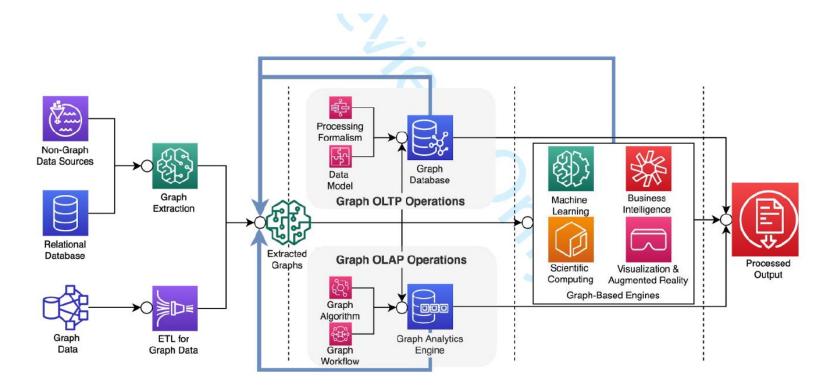
• Several killer applications exist, e.g. financial, logistic, scientific, fraud detection, cybersecurity, supply chain management etc.



Ch1. Expressivity of the graphs/queries

- Dependence on the chosen data model
- How do humans conceptualize graphs?
- The interoperability issues (due to multiple heterogeneous data sources) are to be taken into account
- A data model lattice to navigate across data models, balancing understandability and expressive power
- A new algebra for the variety of graph workloads

Ch2. A complex data management ecosystem



Ch3. Performance and benchmarking

- The need for new, reproducible experimental methodologies to facilitate quick yet meaningful performance-testing?
- How to define more faithful metrics for executing a graph algorithm, query, program, or workflow?
- How to generate workloads with combined operations, covering temporal, spatial, and streaming aspects?
- How to benchmark pipelines including machine learning and simulation?

Dell'Aglio et al

RSP-QL

- Input data: RDF Triples (s,p,o)
- Semantics is based on denotation
 - extends SPARQL with window functions (outside algebraic structure)
 - derived from CQL
- output: time-annotated binding
 - or graphs



```
REGISTER RSTREAM :outStream AS
SELECT ?green
FROM NAMED WINDOW :window ON :colorStream [RANGE PT15S STEP PT5S]
WHERE {
    WINDOW :window {
        ?green a color:Green.
    }
}
```

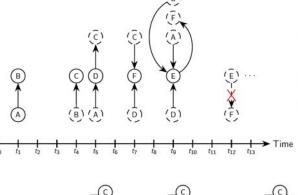
Streaming Graphs

Towards Stream Graph Processing Systems

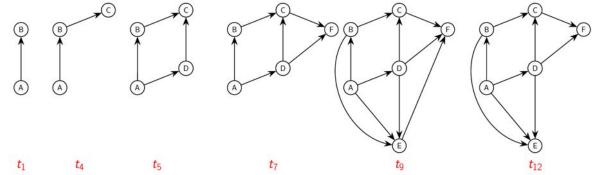
- Dynamic graphs are graphs that can accommodate updates (insertions, deletions, changes) and allow querying on the new/old state
- Streaming graphs are graphs that are unbound as new data arrives at high-speed.
- Current systems and libraries (Gelly/Apache Flink) focus on aggregates/projections
- However, more complex query processing operators taking into account recursion, path-oriented semantics etc. need to be investigated
- Graph processing systems are also inherently dynamic and need to respond to all these challenges

Streaming Graphs

Building the underlying graph one edge at a time

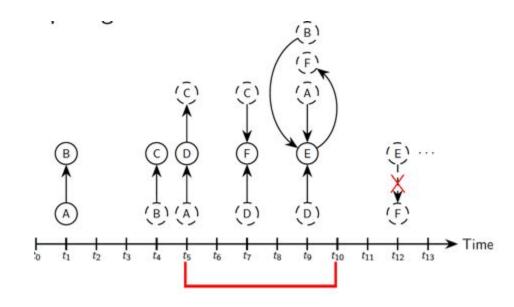


- Combines two difficult problems: streaming+graphs
- Unbounded \Rightarrow don't see entire graph
- Streaming rates can be very high



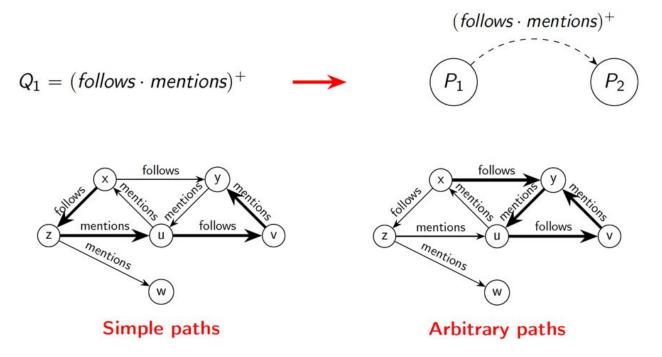
Streaming graph models

- Window-based semantics (use window to batch edges)
- Continuous semantics (edges are batched as they come)
- Complex vs. Simple operations



Streaming Regular Path Queries (RPQs)

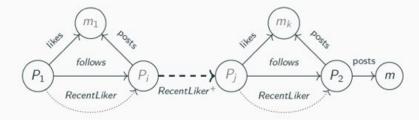
Reflecting the different semantics of graph queries



Anil Pacaci et al.: Regular Path Query Evaluation on Streaming Graphs. SIGMOD Conference 2020: 1415-1430

Towards a Streaming Graph Query Processor

Based on LDBC SNB Interactive Query 7:



G-CORE representation

Datalog program:

 $RL(u_1, u_2) \leftarrow l(u_1, m_1), f(u_1, u_2), p(u_2, m_1)$ Answer $(u, m) \leftarrow RL^+(u, u_2), p(u_2, m)$

Streaming Graph Algebra

A common foundation for streaming graph query engines

Logical query plan G-CORE Query: PATH RL = (x) - [:follows] ->(y), Answer (x)-[:likes]->(m1)<-[:posts]-(y)</pre> CONSTRUCT (p1) -[:notify]-> (m) $\bowtie_{(src1, trg2, notify)}^{\phi_2}$ MATCH (p1)-/ <~RL+> /->(p2), (p2)-[:posts]->(m) \mathcal{W}^{24} $\mathcal{P}_{RLP}^{RL^+}$ ON ldbc_stream WINDOW(24 hours) $\bowtie_{src1,src2,RL}^{\phi_1}$ SGA Expression posts $S_l = \sigma_{l=likes}(\mathcal{W}^{24}(S))$ W^{24} W^{24} W^{24} $S_f = \sigma_{l=follows}(\mathcal{W}^{24}(S))$ $S_p = \sigma_{I=posts}(\mathcal{W}^{24}(S))$ likes posts follows $S_{RecentLiker} = \bowtie_{\phi}^{src1, src3, RecentLiker} (S_{likes}, S_{follows}, S_{posts})$ $S_{Related} = \mathcal{P}_{RecentLiker^+}^{Notify}(S_{RecentLiker})$ Answer = $\bowtie_{\phi}^{src1, trg2, Notify} (S_{Related}, S_p)$

A. Pacaci, A. Bonifati, T. Ozsu. Evaluating Complex Queries on Streaming Graphs. In IEEE ICDE 2022

Streaming graphs in graph query languages

- Input stream of edges (tuples)
- Snapshot graph is the query focus
- Query models is based on non-recursive Datalog + Kleene Star
- Semantics derived from snapshot reducibility (over graphs)
- Output: a graph (path) detected (as in standard query languages, such as GQL and SQL/PGQ)





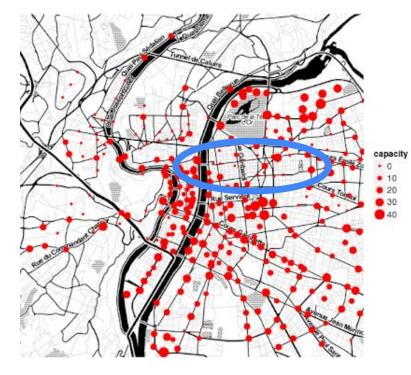
How can we continuously process large graph streams as soon as they are discovered?

Can we design a declarative language that enables continuous graph querying?

Research Question

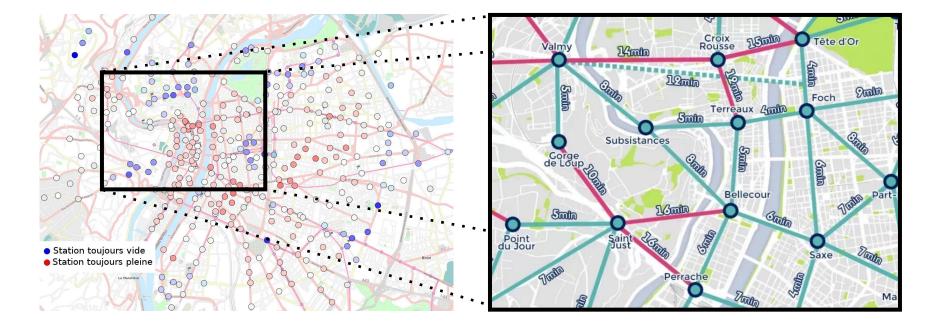
Research Goal

Detecting Free Riders in Smart Biking

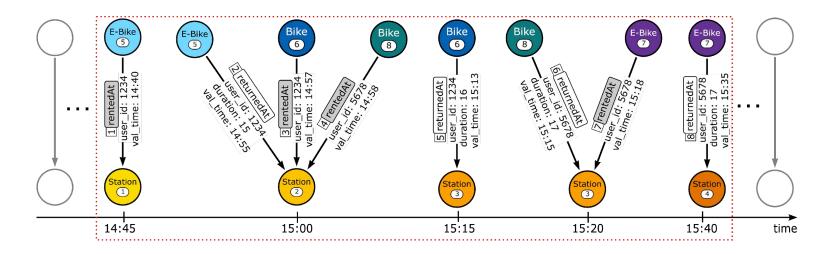




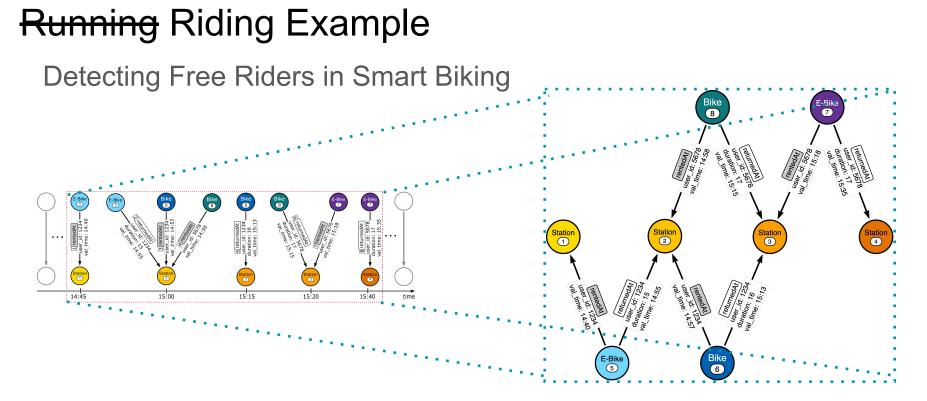
Detecting Free Riders in Smart Biking



Detecting Free Riders in Smart Biking



What users have used the free period for subsequent rentals in the last hour?

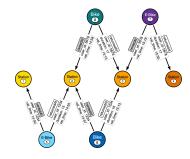


What users have used the free period for subsequent rentals in the last hour?

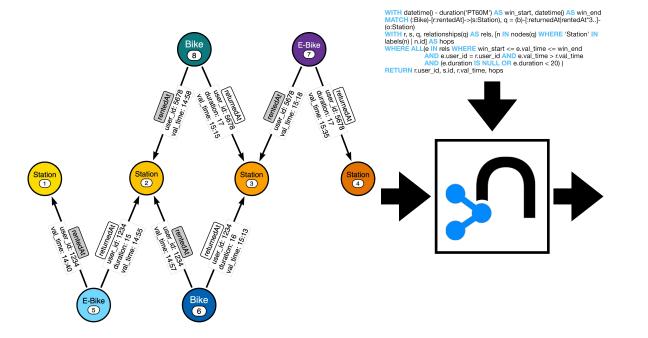
Detecting Free Riders in Smart Biking

What users have used the free period for subsequent rentals in the last hour?

WITH datetime() - duration('PT60M') AS win_start, datetime() AS win_end MATCH (:Bike)-[r:rentedAt]->(s:Station), q = (b)-[:returnedAt|rentedAt*3..]-(o:Station) WITH r, s, q, relationships(q) AS rels, [n IN nodes(q) WHERE 'Station' IN labels(n) | n.id] AS hops WHERE ALL(e IN rels WHERE win_start <= e.val_time <= win_end AND e.user_id = r.user_id AND e.val_time > r.val_time AND (e.duration IS NULL OR e.duration < 20)) RETURN r.user_id, s.id, r.val_time, hops

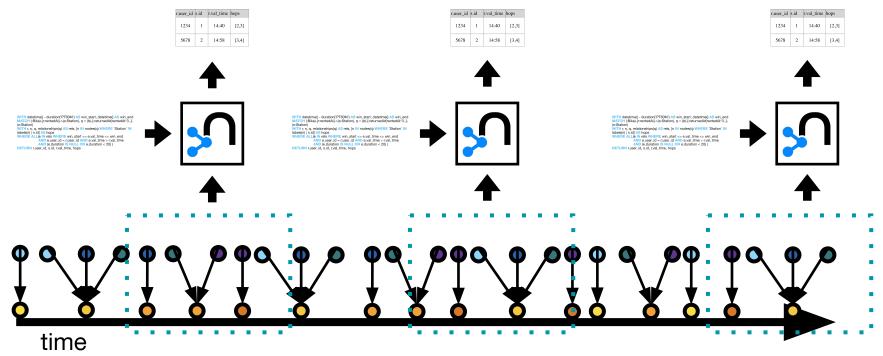


Detecting Free Riders in Smart Biking



r.user_id	s.id	r.val_time	hops
1234	1	14:40	[2,3]
5678	2	14:58	[3,4]

Detecting Free Riders in Smart Biking



Using Cypher for Streaming

Pros

- Declarative
- Interactive
- Intuitive
- Standard-ISH*

- Cons
- •Temporality is reduced to a selection
 - \circ Verbose
 - Unoptimised
- •"Now" = User Time
 - Not Reactive
- •Results Reporting:
 - Now + Latency

You don't really know someone your data until you fight stream them

- **Declarative Semantics**. Seraph allows systems portability and optimisations, as well as adoption.
- **Continuous evaluation.** Seraph's operators allow the repeated evaluation over time, i.e., choosing a time interval and a sequence to evaluate the query.
- **Result emitting**. Seraph's operators allow controlling the report of results, i.e., what is part of the result and when it will be ready to be emitted.
- **Preserving expressiveness**. Seraph preserves openCypher's expressiveness



Seraph's Syntax Before

WITH datetime() - duration('PT60M') AS win start, datetime() AS win end **MATCH** (:Bike)-[r:rentedAt]->(s:Station), q = (b)-[:returnedAt|rentedAt*3..]-(o:Station) WITH r, s, q, relationships(q) AS rels, [n IN nodes(q) WHERE 'Station' IN labels(n) n.id] AS hops WHERE ALL(e IN rels WHERE win start <= e.val time <= win end AND e.user id = r.user id AND e.val time > r.val time AND (e.duration IS NULL **OR** e.duration < 20) **RETURN** r.user_id, s.id, r.val_time, hops

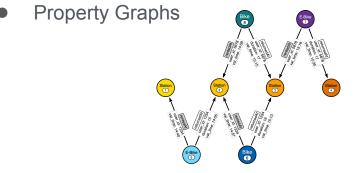
After

REGISTER QUERY student_trick STARTING AT 2022-10-14T14:45 {

MATCH (:Bike)-[r:rentedAt]->(s:Station), q = (b)-[:returnedAt|rentedAt*3..]- (o:Station) WITHIN PT1H WITH r, s, q, relationships(q) AS rels, [n IN nodes(q) WHERE 'Station' IN labels(n) | n.id] AS hops WHERE ALL(e IN rels WHERE e.user_id = r.user_id AND e. val_time > r.val_time AND e.duration < 20) EMIT r.user_id, s.id, r.val_time, hops ON ENTERING EVERY PT5M }

Seraph's Data Model

Before



• Tables

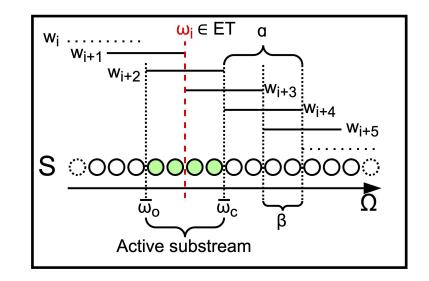
r.user_id	s.id	r.val_time	hops
1234	1	14:40	[2,3]
5678	2	14:58	[3,4]

After

- Property Graph Stream
 - unbounded ordered sequence of pairs (G, ω) where G is a PG and ω a timestamp
- Snapshot Graph
 - Union of all the PGs within a finite sub-portion of a PGStream
- Time-annotated Tables
 - Extend Tables with temporal Bound
- Time-varying Table ($\Psi: \Omega \rightarrow T$)
 - a functional extension of the relational model to incorporate the time semantics

Seraph's Data Model

- Time-varying Table ($\Psi: \Omega \rightarrow T$)
 - Consistency, i.e., Ψ always
 identifies a time-annotated table
 - Chronologicality, i.e., Ψ always identifies the time-annotated table with the earliest (minimal) opening timestamp.
 - Monotonicity, i.e., Ψ always identifies subsequent time-annotated tables for subsequent time instants.



Seraph's Semantics

$$\begin{bmatrix} \mathsf{RETURN} * \end{bmatrix}_{\widetilde{G}}^{\infty}(\Psi, \omega) = \Psi(\omega) \text{ where } \omega \in [\omega_{0}, \omega_{c}), \text{ and } \omega_{0}, \omega_{c} \text{ are the time annotations of } \Psi(\omega) \\ \begin{bmatrix} \mathsf{EMIT} * \end{bmatrix}_{\widetilde{G}}^{\infty}(\Psi, \omega) = \forall \omega_{e} \in ET \begin{bmatrix} \mathsf{RETURN} * \end{bmatrix}_{\widetilde{G}}^{\infty}(\Psi, \omega_{e}) \text{ a proposal} \\ \begin{bmatrix} \mathsf{EMIT} * \mathsf{ON} \mathsf{ENTERING} \end{bmatrix}_{\widetilde{G}}^{\infty}(\Psi, \omega) = \begin{bmatrix} \mathsf{EMIT} * \end{bmatrix}_{\widetilde{G}}^{\infty}(\Psi, \omega), \text{ where } \Psi = \{\mu \mid \mu \in \Psi(\omega) \setminus \Psi(\omega - 1)\} \\ \begin{bmatrix} \mathsf{EMIT} * \mathsf{ON} \mathsf{EXIT} \end{bmatrix}_{\widetilde{G}}^{\infty}(\Psi, \omega) = \begin{bmatrix} \mathsf{EMIT} * \end{bmatrix}_{\widetilde{G}}^{\infty}(\Psi, \omega), \text{ where } \Psi = \{\mu \mid \mu \in \Psi(\omega) \setminus \Psi(\omega) - 1 \setminus \Psi(\omega)\} \\ \begin{bmatrix} \mathsf{EMIT} * \mathsf{SNAPSHOT} \end{bmatrix}_{\widetilde{G}}^{\infty}(\Psi, \omega) = \begin{bmatrix} \mathsf{EMIT} * \end{bmatrix}_{\widetilde{G}}^{\infty}(\Psi, \omega), \text{ where } \Psi = \{\mu \mid \mu \in \Psi(\omega) - 1 \setminus \Psi(\omega)\} \\ \begin{bmatrix} \mathsf{WITH} * \mathsf{SNAPSHOT} \end{bmatrix}_{\widetilde{G}}^{\infty}(\Psi, \omega) = \Psi(\omega) \text{ if } \Psi(\omega) \text{ has at least one field} \\ \begin{bmatrix} \mathsf{WITH} * \mathsf{IH} * \mathbb{I}_{\widetilde{G}}^{\infty}(\Psi, \omega) = \Psi(\omega) \text{ if } \Psi(\omega) \text{ has at least one field} \\ \begin{bmatrix} \mathsf{WITH} \text{ ret } \mathsf{WHERE} \text{ expr} \end{bmatrix}_{\widetilde{G}}^{\infty} = \begin{bmatrix} \mathsf{MATCH} \pi \end{bmatrix}_{S}^{W(\omega_{0},\alpha,\beta)}(\Psi, \omega) \\ \cong \begin{bmatrix} \mathsf{MATCH} \pi \end{bmatrix}_{W(\omega_{0},\alpha,\beta)}(S)(\Psi, \omega) \\ \cong \begin{bmatrix} \mathsf{MATCH} \pi \end{bmatrix}_{\widetilde{S}_{\omega_{c}}^{\infty}(\omega)}(\Psi, \omega) \\ \cong \begin{bmatrix} \mathsf{MATCH} \pi \end{bmatrix}_{\widetilde{S}_{\omega_{c}}^{\infty}(\psi, \omega)} \\ = \forall_{\mu \in \Psi(\omega)} \{\mu \cdot \mu' \mid \mu' \in \overline{\mathsf{match}}(\pi, \widetilde{G}, \mu)\} \\ \end{bmatrix} \begin{bmatrix} \mathsf{SURT}_{\mathsf{VIV}} = \mathsf{Substream} \end{bmatrix}_{\widetilde{S}}^{\mathsf{VIV}} = \mathsf{MUTCH}_{\mathsf{T}}^{\mathsf{VIV}} \otimes \mathsf{Substream}$$

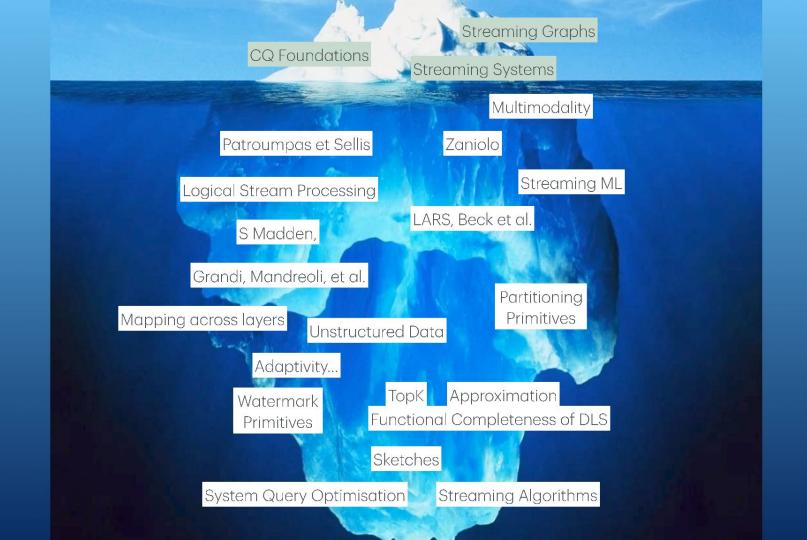
Graph Streaming

is in its infancy

- Need of considering standard graph query languages
- Need of adapting graph query semantics (trail, shortest path etc.)
- Need to make it efficient
- Related problems: quality-aware streaming, fairness-aware streaming



You don't really know someone your graph until you fight stream them



Linguistic Maturity

- what are the the fundamental abstractions to enable continuous queries ?
- In the first "Stream Processing Era", several foundational languages have been presented
 - Terry et al, CQL, Kramer et al. but the list continues.
- Are the language design principles shared?



Query Portability

what are the the fundamental abstractions to enable continuous queries ?

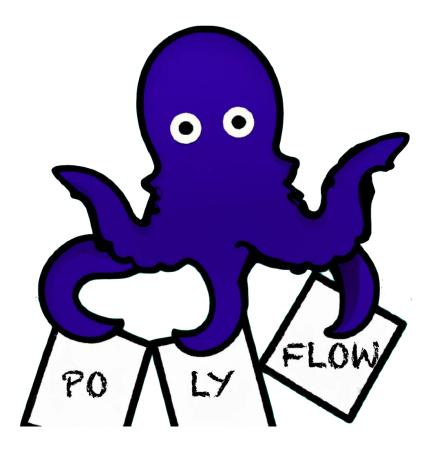
Surprisingly(?), it is not that easy to port continuous queries from a system to another

Intermediate representation like River, DBSP, Brooklet, Arc, go in such direction



Data Complexity

- Users are demanding more and more sophisticated view over data
- In the while data may seems simple, but information needs push for challenging this assumption
- How far can we push such challenge? to what data models SP generalise?
- Can we extend into unstructured multimodality?



Observations

Linguistic Maturity

- "windows" aside, what are the the fundamental abstractions to enable continuous queries ?
- There are evidence of industrial adoption and standardization seems possible
- Despite the variety of language design proposal, such languages are still "domain-specific"

Query Portability

- Streaming System internals remain largely custom, hindering query portability
- Can we dissect the modern and established approaches to uniform them?
- Intermediate representation like River, DBSP, Brooklet, Arc, go in such direction

Data Complexity

- Users are demanding more and more complex view over data
- How far can we push such challenge? to what data models SP generalise?
- How far can we push such challenge? to what data models SP generalise?

Q&A

An Overview of Continuous Querying in (Modern) Data System

Riccardo Tommasini, INSA Lyon, CNRS Liris (France) Angela Bonifati, Lyon 1 University, CNRS Liris, IUF (France)



