Snapshot Semantics for Temporal Multiset Relations

Anton Dignös[1] Boris Glavic[2] Xing Niu[2] Johann Gamper[1]
[3]Michael H. Böhlen

[1]*Free University of Bozen-Bolzano, Italy*

[2]*Illinois Institute of Technology, USA*

[3]*University of Zurich, Switzerland*

VLDB' 19, Los Angeles, USA

Outline

1. Introduction
2. Three Problems
3. Our Approach
4. Experiments
5. Conclusions and Future Work

Background and Motivation

- **Temporal Databases**
  - Record how data changes over time
  - Different query languages, operators and data structures have been proposed
  - Renewed interest from database vendors (temporal features in SQL:2011)

Background and Motivation

- **Temporal Databases**
  - Record how data changes over time
  - Different query languages, operators and data structures have been proposed
  - Renewed interest from database vendors (temporal features in SQL:2011)
- **Snapshot Semantics**
  - Important class of temporal queries
  - Considers a temporal database as a sequence of snapshots
  - Existing approaches in some cases fail to fulfill fundamental correctness criteria

Background and Motivation

- **Temporal Databases**
  - Record how data changes over time
  - Different query languages, operators and data structures have been proposed
  - Renewed interest from database vendors (temporal features in SQL:2011)
- **Snapshot Semantics**
  - Important class of temporal queries
  - Considers a temporal database as a sequence of snapshots
  - Existing approaches in some cases fail to fulfill fundamental correctness criteria
- **We propose . . .**
  - the first **provably correct** approach for snapshot semantics
  - that works for **bags**, sets, and more (e.g., provenance)
  - and is implemented using **SQL period relations**

Snapshot Semantics

Snapshot Semantics

- Evaluates a non-temporal query Q over a temporal database D
- The query is evaluated over each snapshot $\tau_T(D)$
- The result is a temporal database - how Q's answer changes over time

Snapshot Semantics

- Evaluates a non-temporal query Q over a temporal database D
- The query is evaluated over each snapshot $\tau_T(D)$
- The result is a temporal database - how Q's answer changes over time

Definition (Snapshot Reducibility)

$$\tau_T(Q(D)) = Q(\tau_T(D))$$

- Each time point T is associated with the result of the query at this point in time
- Essential correctness criterion for snapshot semantics!

Snapshot Semantics - Example

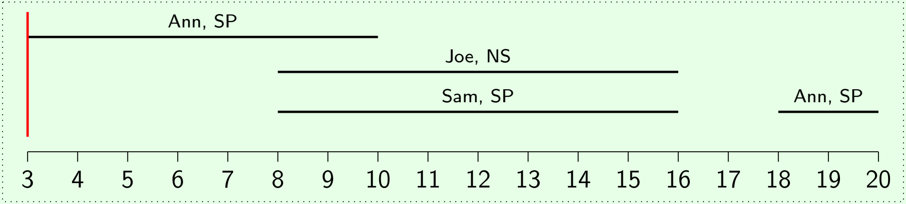
- Q_{onduty}: `SELECT count(*) AS cnt FROM works`

name	skill	period
Ann	SP	[03, 10)
Joe	NS	[08, 16)
Sam	SP	[08, 16)
Ann	SP	[18, 20)

name	skill
Ann	SP

$\xrightarrow{Q_{onduty}}$

cnt
1



Snapshot Semantics - Example

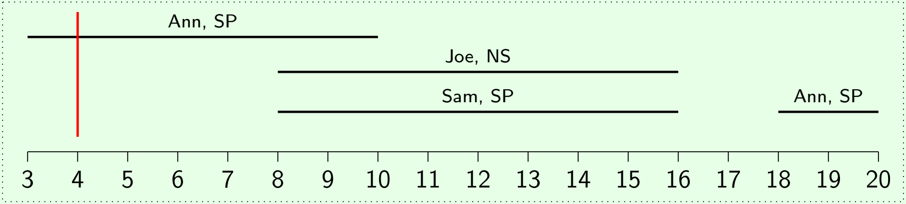
- Q_{onduty}: `SELECT count(*) AS cnt FROM works`

name	skill	period
Ann	SP	$[03, 10)$
Joe	NS	$[08, 16)$
Sam	SP	$[08, 16)$
Ann	SP	$[18, 20)$

name	skill
Ann	SP

$\xrightarrow{Q_{onduty}}$

cnt
1



Snapshot Semantics - Example

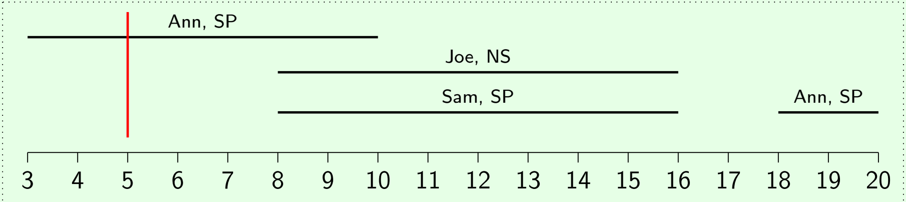
- Q_{onduty}: `SELECT count(*) AS cnt FROM works`

name	skill	period
Ann	SP	$[03, 10)$
Joe	NS	$[08, 16)$
Sam	SP	$[08, 16)$
Ann	SP	$[18, 20)$

name	skill
Ann	SP

$\xrightarrow{Q_{onduty}}$

cnt
1



Snapshot Semantics - Example

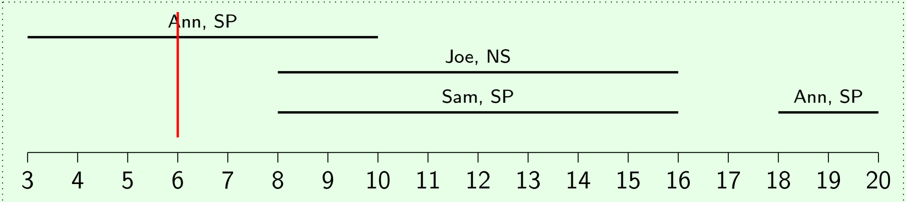
- Q_{onduty}: `SELECT count(*) AS cnt FROM works`

name	skill	period
Ann	SP	[03, 10)
Joe	NS	[08, 16)
Sam	SP	[08, 16)
Ann	SP	[18, 20)

name	skill
Ann	SP

$\xrightarrow{Q_{onduty}}$

cnt
1



Snapshot Semantics - Example

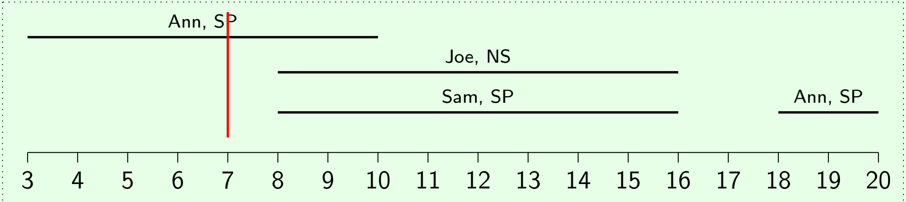
- Q_{onduty}: `SELECT count(*) AS cnt FROM works`

name	skill	period
Ann	SP	[03, 10)
Joe	NS	[08, 16)
Sam	SP	[08, 16)
Ann	SP	[18, 20)

name	skill
Ann	SP

$\xrightarrow{Q_{onduty}}$

cnt
1



Snapshot Semantics - Example

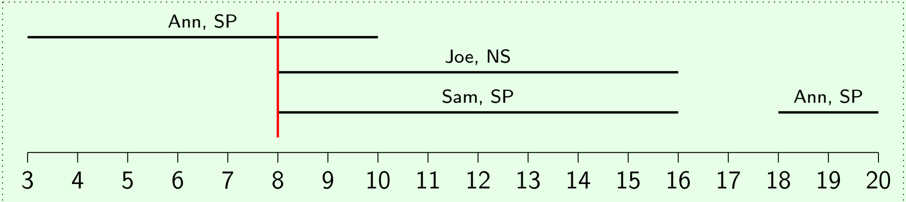
- Q_{onduty}: `SELECT count(*) AS cnt FROM works`

name	skill	period
Ann	SP	$[03, 10)$
Joe	NS	$[08, 16)$
Sam	SP	$[08, 16)$
Ann	SP	$[18, 20)$

name	skill
Ann	SP
Joe	NS
Sam	SP

$\xrightarrow{Q_{onduty}}$

cnt
3



Snapshot Semantics - Example

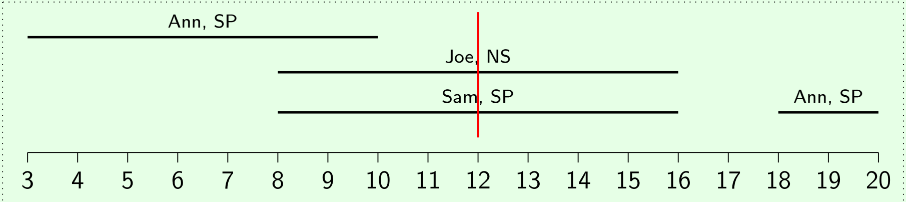
- Q_{onduty}: `SELECT count(*) AS cnt FROM works`

name	skill	period
Ann	SP	$[03, 10)$
Joe	NS	$[08, 16)$
Sam	SP	$[08, 16)$
Ann	SP	$[18, 20)$

name	skill
Joe	NS
Sam	SP

$\xrightarrow{Q_{onduty}}$

cnt
2



Snapshot Semantics - Example

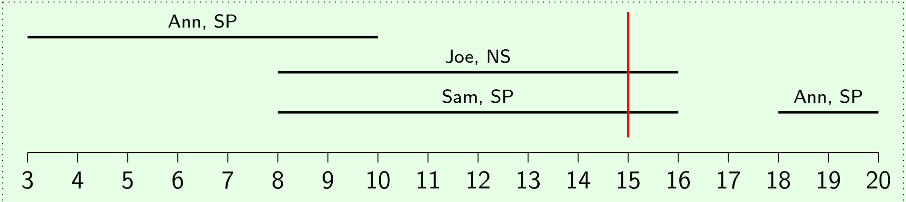
- Q_{onduty}: `SELECT count(*) AS cnt FROM works`

name	skill	period
Ann	SP	$[03, 10)$
Joe	NS	$[08, 16)$
Sam	SP	$[08, 16)$
Ann	SP	$[18, 20)$

name	skill
Joe	NS
Sam	SP

$\xrightarrow{Q_{onduty}}$

cnt
2



Snapshot Semantics - Example

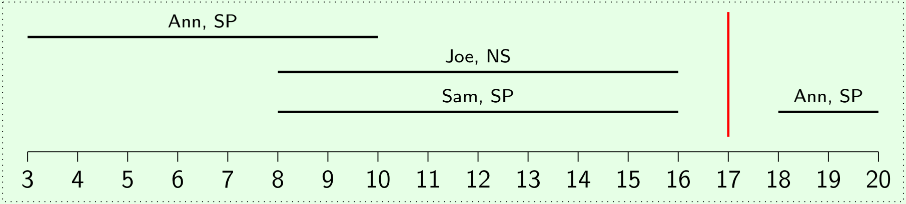
- Q_{onduty}: `SELECT count(*) AS cnt FROM works`

name	skill	period
Ann	SP	$[03, 10)$
Joe	NS	$[08, 16)$
Sam	SP	$[08, 16)$
Ann	SP	$[18, 20)$

name	skill

$\xrightarrow{Q_{onduty}}$

cnt
0



Snapshot Semantics - Example

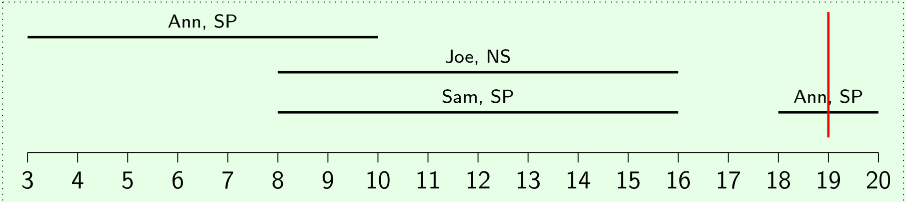
- Q_{onduty}: `SELECT count(*) AS cnt FROM works`

name	skill	period
Ann	SP	[03, 10)
Joe	NS	[08, 16)
Sam	SP	[08, 16)
Ann	SP	[18, 20)

name	skill
Ann	SP

$\xrightarrow{Q_{onduty}}$

cnt
1



Snapshot Semantics - Example

- Q_{onduty}: `SELECT count(*) AS cnt FROM works`
- Merging of snapshot into intervals
- Possible interval encoding of the query result

name	skill	period
Ann	SP	[03, 10)
Joe	NS	[08, 16)
Sam	SP	[08, 16)
Ann	SP	[18, 20)

$\xrightarrow{Q_{onduty}}$

cnt	period
0	[00, 03)
1	[03, 08)
3	[08, 10)
2	[10, 16)
0	[16, 18)
1	[18, 20)
0	[20, 23)

Outline

Problem I: Aggregation Gap (AG) Bug

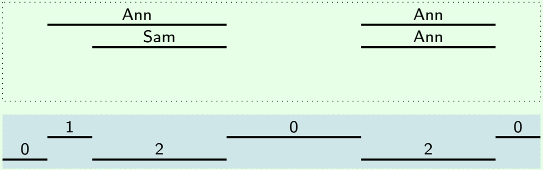
- Q_{duty}: `SELECT count(*) AS cnt FROM works`

works

name	period
Ann	$[01, 05)$
Sam	$[02, 05)$
Ann	$[08, 11)$
Ann	$[08, 11)$

$\xrightarrow{Q_{AG}}$

cnt	period
0	$[00, 01)$
1	$[01, 02)$
2	$[02, 05)$
0	$[05, 08)$
2	$[08, 11)$
0	$[11, 12)$



- No approach correctly handles gaps for aggregation!

  $\rightarrow$ Violation of snapshot reducibility!

Problem II: Bag Difference (BD) Bug

- Q_{BD}: `SELECT name FROM assign EXCEPT ALL SELECT name FROM works`

assign

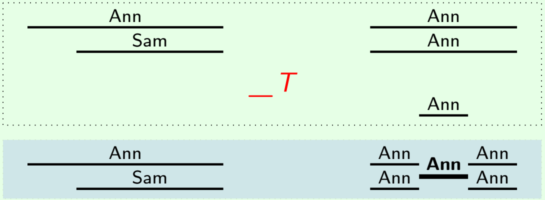
name	period
Ann	[00, 04)
Sam	[01, 04)
Ann	[07, 10)
Ann	[07, 10)

works

name	period
Ann	[8, 9)

$\xrightarrow{Q_{BD}}$

name	period
Ann	[00, 04)
Sam	[01, 04)
Ann	[07, 08)
Ann	[07, 08)
Ann	[08, 09)
Ann	[09, 10)
Ann	[09, 10)

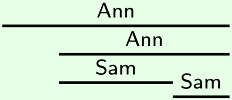


- Most approaches perform a `NOT EXISTS`
  → Violation of snapshot reducibility!

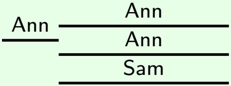
Problem III: Unique Interval Encoding

- **What?**
  - Snapshot reducibility only tells us snapshots of the result

name	period
Ann	[00, 04)
Ann	[01, 04)
Sam	[01, 03)
Sam	[03, 04)



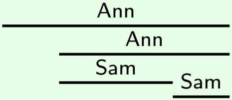
name	period
Ann	[00, 01)
Ann	[01, 04)
Ann	[01, 04)
Sam	[01, 04)



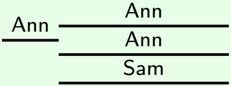
Problem III: Unique Interval Encoding

- **What?**
  - Snapshot reducibility only tells us snapshots of the result
- **Why uniqueness?**
  - Equivalence rules hold, eg., $r \cap s \equiv r - (r - s)$

name	**period**
Ann	$[00, 04)$
Ann	$[01, 04)$
Sam	$[01, 03)$
Sam	$[03, 04)$



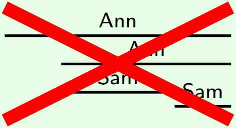
name	**period**
Ann	$[00, 01)$
Ann	$[01, 04)$
Ann	$[01, 04)$
Sam	$[01, 04)$



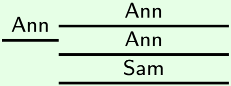
Problem III: Unique Interval Encoding

- **What?**
  - Snapshot reducibility only tells us snapshots of the result
- **Why uniqueness?**
  - Equivalence rules hold, eg., $r \cap s \equiv r - (r - s)$
- **How?**
  - Generalized coalescing

name	**period**
Ann	$[0[illegible], 04)$
Ann	$[01, 04)$
Sam	$[01, 03)$
Sam	$[03, 04)$



name	**period**
Ann	$[00, 01)$
Ann	$[01, 04)$
Ann	$[01, 04)$
Sam	$[01, 04)$



Related Work

Approach	Bag	AG bug free	BD bug free	Unique encoding
Interval preservation [Böhlen et al., 2000] (ATSQL)	✓	×	×	×
Teradata [Teradata, 2015]	✓	×	N/A	×[a]
Change preservation [Dignös et al., 2012, Dignös et al., 2016]	×	×	N/A	×
TSQL2 [Snodgrass, 1995, Snodgrass et al., 1994, Soo et al., 1995]	×	N/A	N/A	✓
ATSQL2 [Böhlen et al., 1995]	✓	N/A	×	×
TimeDB [Steiner, 1998] (ATSQL2)	✓	N/A	×	×
SQL/Temporal [Snodgrass et al., 1996]	✓	×	×	×
SQL/TP [Toman, 1998][b]	✓	✓	✓	×
Our approach	✓	✓	✓	✓

[a] Optionally, coalescing (`NORMALIZE ON` in Teradata) can be applied to get a unique encoding at the cost of loosing multiplicities.

[b] Sequenced semantics can be expressed, but this is inefficient

Requirements

Approach for snapshot semantics that ...

- supports bags, sets, and more
- provably snapshot-reducible
- unique encoding
- can be implemented for SQL period relations

Contributions

- **First provably correct approach for snapshot semantics over multisets**
  - Based on **semiring annotated databases** (treat time as annotations)
  - Supports also set semantics and provenance, incompleteness, ...
  - Supports expressive query language (full relational algebra + aggregation)

Contributions

- **First provably correct approach for snapshot semantics over multisets**
  - Based on **semiring annotated databases** (treat time as annotations)
  - Supports also set semantics and provenance, incompleteness, ...
  - Supports expressive query language (full relational algebra + aggregation)
- **Unique encoding**
  - Through **generalized coalescing**

Contributions

- **First provably correct approach for snapshot semantics over multisets**
  - Based on **semiring annotated databases** (treat time as annotations)
  - Supports also set semantics and provenance, incompleteness, ...
  - Supports expressive query language (full relational algebra + aggregation)
- **Unique encoding**
  - Through **generalized coalescing**
- **Implementation**
  - Supports **set and bag semantics** over **SQL period relations**

Outline

Bags as Semirings

- We employ semirings to annotate temporal relations

Bags as Semirings

- We employ semirings to annotate temporal relations
- Bag semiring $(\mathbb{N}, +, \cdot, 0, 1)$

Bags as Semirings

- We employ semirings to annotate temporal relations
- Bag semiring $(\mathbb{N}, +, \cdot, 0, 1)$

name	$\mathbb{N}$
Ann	1
Sam	2

$\cup$

name	$\mathbb{N}$
Ann	1
Joe	1

$=$

name	$\mathbb{N}$
Ann	$1 + 1$
Sam	2
Joe	1

Bags as Semirings

- We employ semirings to annotate temporal relations
- Bag semiring $(\mathbb{N}, +, \cdot, 0, 1)$

name	$\mathbb{N}$
Ann	1
Sam	2

$\cup$

name	$\mathbb{N}$
Ann	1
Joe	1

$=$

name	$\mathbb{N}$
Ann	$1 + 1$
Sam	2
Joe	1

name	$\mathbb{N}$
Ann	1
Sam	2

$\bowtie$

name	$\mathbb{N}$
Ann	1
Joe	1

$=$

name	$\mathbb{N}$
Ann	$1 \cdot 1$

unibz

Bags as Semirings

- We employ semirings to annotate temporal relations
- Bag semiring $(\mathbb{N}, +, \cdot, 0, 1)$

name	$\mathbb{N}$
Ann	1
Sam	2

$\cup$

name	$\mathbb{N}$
Ann	1
Joe	1

$=$

name	$\mathbb{N}$
Ann	$1 + 1$
Sam	2
Joe	1

name	$\mathbb{N}$
Ann	1
Sam	2

$\bowtie$

name	$\mathbb{N}$
Ann	1
Joe	1

$=$

name	$\mathbb{N}$
Ann	$1 \cdot 1$

- Aggregation based on using symbolic expression for values [Amsterdamer et al., 2011]
- Difference based on monus [Geerts and Poggi, 2010]

Running Example

- Relation "works" with factory workers and specializations

name	skill	period
Ann	SP	$[03, 10)$
Joe	NS	$[08, 16)$
Sam	SP	$[08, 16)$
Ann	SP	$[18, 20)$

- Number of specialized workers in the company

Q_{onduty}:

```
SELECT count(*) AS cnt
FROM works
WHERE skill = 'SP'
```

Approach - In a Nutshell

Snapshot K-relations

Abstract

00 ↦

name	**skill**	$\mathbb{N}$

...

08 ↦

name	**skill**	$\mathbb{N}$
Ann	SP	1
Joe	NS	1
Sam	SP	1

...

18 ↦

name	**skill**	$\mathbb{N}$
Ann	SP	1

...

... Q_{onduty} ⟶ ... Q_{onduty} ⟶ ...

00 ↦

cnt	$\mathbb{N}$
0	1

...

08 ↦

cnt	$\mathbb{N}$
2	1

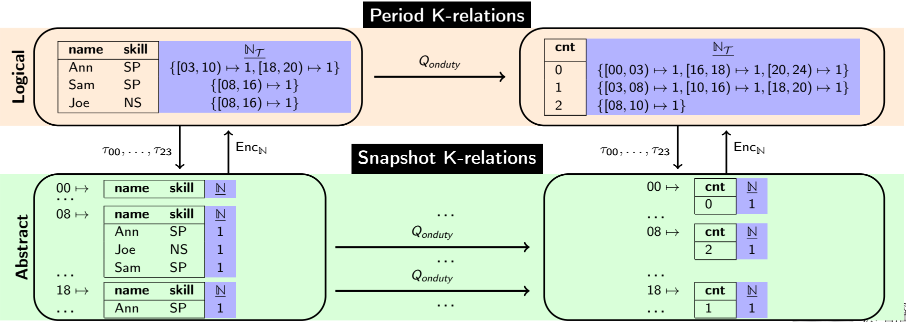
...

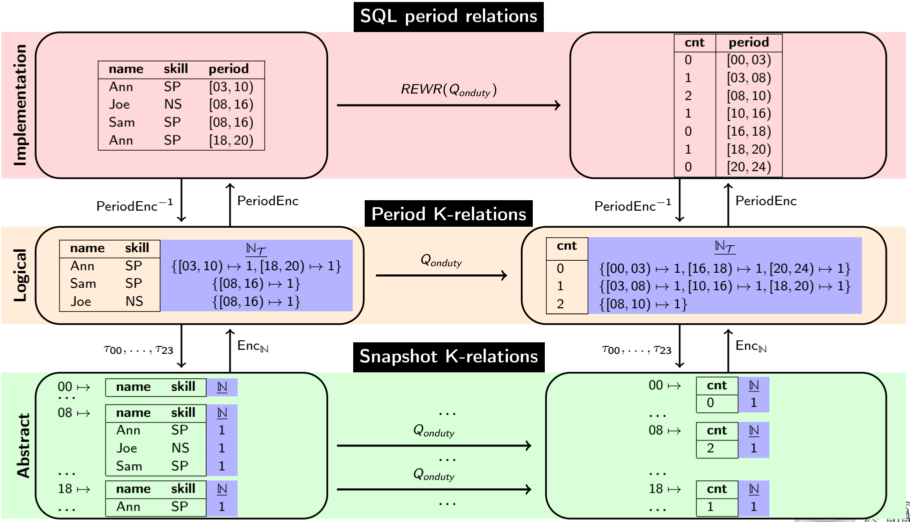
18 ↦

cnt	$\mathbb{N}$
1	1

...

Approach - In a Nutshell



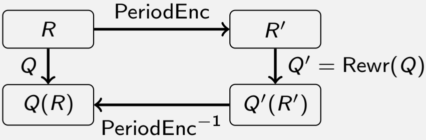

SQL period relations
Implementation
name skill period
Ann SP [03, 10)
Joe NS [08, 16)
Sam SP [08, 16)
Ann SP [18, 20)
REWR(Q_onduty)
cnt period
0 [00, 03)
1 [03, 08)
2 [08, 10)
1 [10, 16)
0 [16, 18)
1 [18, 20)
0 [20, 24)
PeriodEnc^-1
PeriodEnc
Period K-relations
Logical
name skill ℕ_T
Ann SP {[03, 10) ↦ 1, [18, 20) ↦ 1}
Sam SP {[08, 16) ↦ 1}
Joe NS {[08, 16) ↦ 1}
Q_onduty
cnt ℕ_T
0 {[00, 03) ↦ 1, [16, 18) ↦ 1, [20, 24) ↦ 1}
1 {[03, 08) ↦ 1, [10, 16) ↦ 1, [18, 20) ↦ 1}
2 {[08, 10) ↦ 1}
τ00, . . . , τ23
Enc_ℕ
Snapshot K-relations
Abstract
00 ↦ name skill ℕ
08 ↦ name skill ℕ
Ann SP 1
Joe NS 1
Sam SP 1
18 ↦ name skill ℕ
Ann SP 1
Q_onduty
Q_onduty
00 ↦ cnt ℕ
0 1
08 ↦ cnt ℕ
2 1
18 ↦ cnt ℕ
1 1

Implementation

Approach

- **Encode** period $\mathbb{N}$-relations as SQL period relations
- **Rewrite** queries with period $\mathbb{N}$-semantics into SQL

SQL Rewriting



Optimization

- Elimination of redundant coalescing steps
- Optimized rewrites for individual operators

Outline

Setup

- Systems:
  - **PG:** a version of Postgres (PG) **with** native support for temporal operators
  - **DBX:** commercial DBMS **with** native support for snapshot semantics
  - **DBY:** a commercial DBMS, DBY **without** native support for snapshot semantics
- Methods:
  - **Seq:** used our approach to translate snapshot queries into standard SQL queries
  - **Nat:** ran the queries also with the native solution for snapshot semantics paired with our implementation of coalescing to produce a coalesced result
- Datasets:
  - **TPC-BiH:** the bi-temporal version of the TPC-H benchmark dataset (1GB and 10GB)
  - **Employee:** contains $\approx$ 4 million records and consists of six period tables
- Workloads:
  - **TPC-BiH:** 9 of the 22 standard TPC-BiH queries without nested subqueries or LIMIT
  - **Employee:** 10 queries over the employee dataset (join queries, aggregation queries and difference queries)

Sequenced Query Performance

TPC-BiH (10GB)

Query	Q1	Q5	Q6	Q7	Q8	Q9	Q12	Q14	Q19
PG-Seq	63.85	5.85	7.70	28.70	21.78	129.01	10.49	26.55	9.60
PG-Nat	TO (2h)	1794.10	126.91	1642.20	1484.61	TO (2h)	264.57	3436.30	2873.13

Employee

Query	join-1	join-2	join-3	join-4	agg-1	agg-2	agg-3	agg-join	diff-1	diff-2
PG-Seq	91.97	1543.81	0.01	0.52	7.02	0.06	1.42	6643.61	14.18	63.58
PG-Nat	118.01	1543.81	4.91	12.85	5980.85	10.31	0.02	19195.03	6.88	79.63

- In the paper we . . .
  - compare against additional systems
  - evaluate performance of multiset coalescing

Outline

Conclusions

- We present the first provably correct realization of snapshot semantics for multiset relations
- Our solution is based on semiring-annotated data

  $\Rightarrow$ it also applies to sets, provenance, probabilistic data, ...
- Implementation as a rewriting-frontend
  - Applies to data stored as SQL period relations
  - Run on-top of a standard DBMS

Future Work

- Native implementation of K-coalescing
- Extensions for multiple time dimensions
- Study applicability to broader classes of temporal queries

Questions?

Thank you for your attention!

- **Webpage:**
  - http://www.cs.iit.edu/~dbgroup/projects/tempdb.html
- **Github:**
  - https://github.com/IITDBGroup/gprom
- **SQL syntax:**
  - https://github.com/IITDBGroup/gprom/wiki/temporal

Outline

6 Bibliography

[Amsterdamer et al., 2011] Amsterdamer, Y., Deutch, D., and Tannen, V. (2011).
Provenance for aggregate queries.
In *PODS*, pages 153–164.

[Böhlen et al., 1995] Böhlen, M. H., Jensen, C. S., and Snodgrass, R. T. (1995).
Evaluating and enhancing the completeness of tsql2.
Technical Report TR 95-5, Computer Science Department, University of Arizona.

[Böhlen et al., 2000] Böhlen, M. H., Jensen, C. S., and Snodgrass, R. T. (2000).
Temporal statement modifiers.
ACM Trans. Database Syst., 25(4):407–456.

[Dignös et al., 2012] Dignös, A., Böhlen, M. H., and Gamper, J. (2012).
Temporal alignment.
In *SIGMOD*, pages 433–444.

[Dignös et al., 2016] Dignös, A., Böhlen, M. H., Gamper, J., and Jensen, C. S. (2016).
Extending the kernel of a relational DBMS with comprehensive support for sequenced temporal queries.
ACM Trans. Database Syst., 41(4):26:1–26:46.

[Geerts and Poggi, 2010] Geerts, F. and Poggi, A. (2010).
On database query languages for K-relations.
Journal of Applied Logic, 8(2):173–185.

[Snodgrass, 1995] Snodgrass, R. T., editor (1995).
The TSQL2 Temporal Query Language.

References II

[Snodgrass et al., 1994] Snodgrass, R. T., Ahn, I., Ariav, G., Batory, D. S., Clifford, J., Dyreson, C. E., Elmasri, R., Grandi, F., Jensen, C. S., Käfer, W., Kline, N., Kulkarni, K. G., Leung, T. Y. C., Lorentzos, N. A., Roddick, J. F., Segev, A., Soo, M. D., and Sripada, S. M. (1994).
TSQL2 language specification.
SIGMOD Record, 23(1):65–86.

[Snodgrass et al., 1996] Snodgrass, R. T., Böhlen, M. H., Jensen, C. S., and Steiner, A. (1996).
Adding valid time to sql/temporal.
ANSI X3H2-96-501r2, ISO/IEC JTC, 1.

[Soo et al., 1995] Soo, M. D., Jensen, C. S., and Snodgrass, R. T. (1995).
An algebra for tsql2.
In *The TSQL2 temporal query language*, pages 505–546.

[Steiner, 1998] Steiner, A. (1998).
A generalisation approach to temporal data models and their implementations.
PhD thesis, ETH Zurich.

[Teradata, 2015] Teradata (2015).
Teradata database - temporal table support.
http://www.info.teradata.com/download.cfm?ItemID=1006923.

[Toman, 1998] Toman, D. (1998).
Point-based temporal extensions of SQL and their efficient implementation.
In *Temporal databases: research and practice*, pages 211–237.